# Sistemas Informáticos
# Curso 2002-03

---

# *- PROYECTO ARMA -*
# *Herramienta de evaluación de contenidos*


Rafael De Benito Cañizares
Mónica Cortés Guitián
Mª José Montero Llopis
Mª Almudena Mozas Noguerado

Dirigido por:
Prof. Isabel Pita Andreu
Dpto. Sistemas Informáticos y Programación


---


Facultad de Informática
Universidad Complutense de Madrid

Se autoriza a la Universidad Complutense de Madrid a difundir y utilizar con fines académicos, no comerciales, tanto la propia memoria, como el código, la documentación y/o el prototipo desarrollado del proyecto *ARMA* (Herramienta de evaluación de contenidos) realizado para la asignatura de Sistemas Informáticos durante el curso académico 2002-2003, por parte de los alumnos:

Rafael De Benito Cañizares

Mónica Cortés Guitián

Mª José Montero Llopis

Mª Almudena Mozas Noguerado

# ÍNDICE


**1.- Resumen ..... Pág.1**

**2.- Summary ..... Pág.2**

**3.- Instalación del sistema ..... Pág.3**

3.1.- Envío de prácticas a través de Internet ..... Pág.3
3.1.1.- Windows 95/98/Me/2000 ..... Pág.3
3.1.2.- Windows NT ..... Pág.5

3.2.- Herramienta de evaluación de contenidos ..... Pág.5

**4.- Uso de la herramienta ..... Pág.6**

4.1.- Envío de prácticas a través Internet ..... Pág.6

4.2.- Herramienta de evaluación de contenidos ..... Pág.11
4.2.1.- Pantallas de generación de los datos ..... Pág.12
4.2.1.1.- Generación automática de los datos ..... Pág.12
4.2.1.2.- Generación manual de los datos ..... Pág.15
4.2.2.- Pantalla de ejecución de los programas de los alumnos ..... Pág.16
4.2.2.1.- Ejecuciones ..... Pág.16
4.2.2.2.- Comparación de los resultados ..... Pág.23
4.2.3.- Comparación de la estructura de los programas ..... Pág.35
4.2.4.- Listados ..... Pág.39
4.2.4.1.- Generación de listados ..... Pág.39
4.2.4.2.- Generación de estadísticas ..... Pág.50

**5.- Codificación de la herramienta ..... Pág.57**

5.1.- Envío de prácticas a través Internet ..... Pág.57

5.2.- Herramienta de evaluación de contenidos ..... Pág.57
5.2.1.- Generación de los datos ..... Pág.58
5.2.1.1.- Clase PanelGenAutomatica() ..... Pág.58
5.2.1.2.- Clase PanelGenManual() ..... Pág.60
5.2.2.- Ejecución de los programas ..... Pág.64
5.2.2.1.- Clase PanelEjecucion() ..... Pág.64
5.2.3.- Comparación de la estructura de los programas ..... Pág.68
5.2.3.1.- Clase PanelComparacion() ..... Pág.68

**6.- Resultados de las pruebas ..... Pág.80**

6.1.- Pruebas de ASCII ..... Pág.80
6.1.1.- Ejecución de los *.exe* de los alumnos ..... Pág.80
6.1.2.- Compilación de las prácticas ..... Pág.85
6.1.3.- Ejecución de los *.exe* generados por el código de los alumnos ..... Pág.86
6.1.4.- Comparación de los ejecutables dejados y los ejecutables generados ..... Pág.91
6.1.5.- Comparación de los ejecutables con la práctica del profesor ..... Pág.93
6.1.6.- Comparación de los ejecutables generados con la práctica del profesor ..... Pág.96
6.1.7.- Comparación de las estructuras de los programas con la práctica del profesor ..... Pág.98

6.2.- Pruebas de MCD ..... Pág.99
6.2.1.- Generación manual ..... Pág.100
6.2.2.- Generación automática ..... Pág.103

6.3.- Pruebas de MÉTODO DE ORDENACIÓN DE LA BURBUJA ..... Pág.106
6.3.1.- Generación manual ..... Pág.108
6.3.2.- Generación automática ..... Pág.115

**7.- Apéndice I: Código de las prácticas ..... Pág.119**
7.1.- ASCII ..... Pág.119
7.2.- MCD ..... Pág.126
7.3.- BURBUJA ..... Pág.133

**8.- Lista de palabras clave ..... Pág.137**

**9.- Bibliografía ..... Pág.138**

# 1.- RESUMEN

*ARMA* es una herramienta de evaluación de contenidos. Permite al profesor la evaluación y control de programas simples en Pascal, tanto en cuanto al estilo de programación como al correcto funcionamiento de los mismos. *ARMA* está dividido en dos módulos:

- Envío por parte de los alumnos de sus prácticas a través de Internet.
- Evaluación de las prácticas por parte del profesor.

- **Envío de prácticas por Internet:**

Para el envío de las prácticas vía Internet, el alumno debe conectarse a una página web determinada y siguiendo las instrucciones ir enviando sus ficheros debidamente formateados. Se pueden enviar tanto ficheros fuente (*.pas*) como ficheros ejecutables(*.exe*). Los ficheros quedan almacenados en directorios ordenados según las prácticas para el posterior análisis del profesor. A la hora del envío el programa comprueba mediante una lista de alumnos que el alumno que envía los datos existe y su pertenencia a un grupo y curso, así como el correcto formato del fichero. En caso de que haya alguna diferencia entre los datos proporcionados por el alumno y la lista de alumnos se genera un archivo de advertencia para el profesor indicando las diferencias observadas.

- **Evaluación de las prácticas:**

Para la evaluación de las prácticas, el profesor dispone de las siguientes opciones:

- Generación automática de datos de prueba para las prácticas. Se permite al profesor seleccionar cuántos ficheros de prueba desea crear y con qué tipo de datos.
- Generación manual de datos por el profesor. Se le permite introducir los ficheros de datos que él desee.
- Compilación de las prácticas entregadas por los alumnos. Se establecerá si las prácticas entregadas compilan o no.
- Ejecución de los archivos ejecutables(*.exe*) entregados por los alumnos. Se ejecutará el archivo entregado por el alumno y se analizarán los resultados obtenidos.
- Ejecución de los archivos con código fuente(*.pas*) entregados por los alumnos. Se creará el archivo ejecutable(*.exe*) y se comparará si el archivo ejecutable entregado por el alumno coincide con el generado por el sistema, además se comparan los resultados obtenidos tras las ejecuciones de ambos con datos generados automáticamente o con datos proporcionados por el profesor.
- Comparación sintáctica de dos programas en código fuente. Permite evaluar el estilo de programación de un alumno, en relación con el del profesor, o con el de otro compañero a fin de poder comprobar si hay prácticas copiadas. Se tendrá en cuenta la equivalencia de instrucciones como el *WHILE* y el *FOR* por ejemplo.
- Listados de salida. Los datos que se proporcionan para cada alumno son: si la práctica es correcta o no, si compila y si ejecuta, los datos de entrada para los que ha fallado un programa, listados con la comparación de las prácticas y estadísticas sobre los informes anteriores.

# 2.- SUMMARY

*ARMA* is a program evaluating system. It allows to evaluate and examine simple students Pascal programs. *ARMA* is divided in two parts:

- Programs sended by students via internet.
- User programs evaluation.

- **Programas sended by students via Internet:**

The system provides a website used by the students to send their programs. Students can send source files and exe files. The files will be stored in a directory structure, for it's later analysis. During the sending process, *ARMA* checks if the student who is sending programs exists in the teachers database, and his group and year, as well as the file format. If there's any difference between student data and the user's one, a warning file is created in order to tell the teacher the existing differences.

- **User programs evaluation**

The program offers the following options:

- Automatic data generation to test the programs . The user can choose the number of files to be created and the data types.
- Manual data generation. The user can select the files to run the test.
- Programs compilation. *ARMA* can determine if the sent programs compile or not.
- Files execution. *ARMA* will execute files and analize the results.
- Source files compilation and execution. *ARMA* will create the object file, and will compare the created file with the sent one. It will also compare the results given by both files.
- Execution of one single program with specific input data.
- Syntactic comparison of two source files. It allows to evaluate the student programming style in comparising it with the user's one, and to determine if two students have copied by comparing two student programs.
- Listings. The system shows if the program compiles, if it runs, if it works properly, if it has errors, input data for which the program has failed, lists with the program comparation result and statistics on the data above for each student program.

# 3.- INSTALACIÓN DEL SISTEMA

## 3.1.- ENVÍO DE PRÁCTICAS A TRAVÉS DE INTERNET

### 3.1.1.- Windows 95/98/Me/2000

- Se requiere:

  - jdk 1.1.8 máquina virtual java.

  - jswdk-1.0.01 servidor de aplicaciones.

- Configuración del entorno:

  - Editar el archivo *'autoexec.bat'* y en SET PATH añadir *c:\jdk1.1.8\bin*, y añadir al CLASSPATH (en caso de que exista, o crearlo sino existe con SET CLASSPATH) los siguientes archivos: c:\JSWDK-1.0.1\LIB\classes.zip;c:\jdk1.1.8\lib\classes.zip

  - NOTA: ha de tenerse en cuenta la ruta del jswdk, en cada caso donde se haya instalado.

  - Si el archivo 'classes.zip' no se encontrara en la carpeta LIB del jswdk, copiarla de la carpeta lib del jdk.

  - Editar el archivo 'startserver.bat' (Se encuentra dentro de la carpeta jsdwk-1.0.1). Añadir; %c:\jdk1.8.1\Lib\tools.jar% en la línea set CLASSPATH="....." de manera que quede incluido junto con el resto del CLASSPATH, como por ejemplo:
    set CLASSPATH=%appClassPath%;%sysJars%;%C:\jdk1.8.1\Lib\tools.jar%

  - Ejecutar el archivo 'startserver.bat'. Nota: (si saliera una ventana de MS-DOS con el mensaje 'sin espacio en entorno' ir a las propiedades de la ventana de MS-DOS, y cambiar en la pestaña de memoria, el entorno inicial, poner 2816. Si la primera vez no se guardan los cambios hacerlo sobre la propia ventana de MS-DOS). El servidor quedará arrancado.

  - Abrir la ventana del navegador, y escribir http://localhost:8080 debe encontrar la página de inicio.

  - Si se abriera automáticamente una carpeta que se llama JAVA, renombrarla (ya que el servidor utiliza esa carpeta por defecto) y se vuelve a repetir la operación (abrir el navegador y escribir http://localhost:8080). En el monitor se mostrará:
    JavaServer (tm) Web Development Kit
    Version 1.0
    You are viewing the default JSWDK home page which is distributed
    .....

▪ Creación de las carpetas para almacenar la información:

- Crear una carpeta llamada ‘form’ en el directorio C:\jswdk1_0_1\examples\WEB-INF\jsp\beans

- Crear otra carpeta llamada ‘com’ en el directorio C:\jswdk1_0_1\examples\WEB-INF\jsp\beans , para subir los archivos.
  Copiar los archivos formulario.java, formulario.class, formulario2.java y formulario2.class en la carpeta form.

- Crear en la carpeta 'com' una carpeta llamada 'jspsmart', y en ésta crear otra carpeta que se llame 'upload', en esta última copiar los cinco archivos siguientes:

  -File.class
  -Files.class
  -Request.class
  -SmartUpload.class
  -SmartUpoladException.class

- Crear en C:\jswdk1_0_1\webpages una carpeta que se llame 'upload' (ahí irán los Archivos que suban los alumnos).

- Además hay que copiar en esta carpeta la lista de alumnos

- Copiar en C:\jswdk1_0_1\webpages (--> aquí van las paginas web) los siguientes archivos:

  - fondo.gif
  - paginaprincipal.html
  - paso2.jsp
  - resultado.jsp

- Crear en C:/jswdk1_0_1/webpages/upload/ una carpeta que se llame warnings.
  En esta carpeta es donde se guardan los warnings que se generan de los alumnos que introducen datos 'incorrectos' con un nombre de archivo del tipo siguiente:
  p*NºPrac*n*NºDni*.txt

- Crear en C:/jswdk1_0_1/webpages/upload/ una carpeta para cada número de práctica de la que los alumnos vayan a subir sus prácticas con el formato /practica*NºPrac*/

- Para arrancar la aplicación introducir la siguiente dirección en el navegador :
  http://localhost:8080/paginaprincipal.html

### 3.1.2.- Windows NT

Para establecer el PATH correcto de las variables de entorno, ir al panel de control, sistema, entorno y añadirlas en las variables de usuario y variables de sistema.

Si se está usando el JDK 1.2, además se debe:

- Añadir el archivo "tools.jar" que se encuentra en la carpeta lib del jdk, al CLASSPATH
- Establecer como path de la variable JAVA_HOME la ruta de acceso al jdk

## 3.2.- HERRAMIENTA DE EVALUACIÓN DE CONTENIDOS

- Se requiere:
  - Instalación de JBuilder 7 Enterprise
- Ejecución de la aplicación:
  - Abrir el archivo *Proyecto.JPX* (se abrirá por defecto el entorno de la aplicación JBuilder 7 Enterprise).
  - Seleccionar en el menú *Project*, la opción *Run Project.*

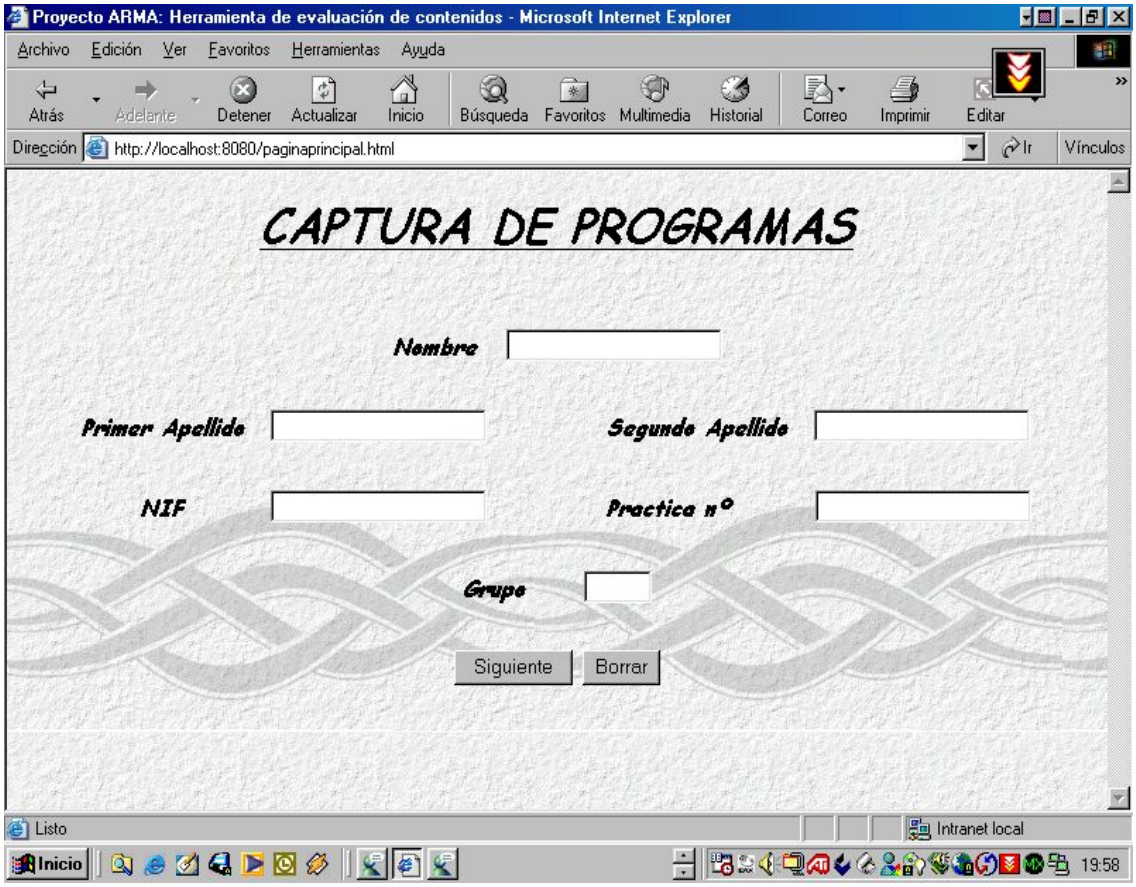
# 4.- USO DE LA HERRAMIENTA

## 4.1.- ENVÍO DE PRÁCTICAS A TRAVÉS DE INTERNET

Dada la simplicidad de las operaciones a realizar en esta parte se ha seleccionado el JSWDK-1.0.1. Un servidor de aplicaciones bastante sencillo, de libre distribución (www.sun.com) que consume pocos recursos y de fácil instalación.

Para acceder a la página desde la que poder enviar los ficheros, se debe introducir la siguiente URL: http://localhost:8080/PaginaPrincipal.html

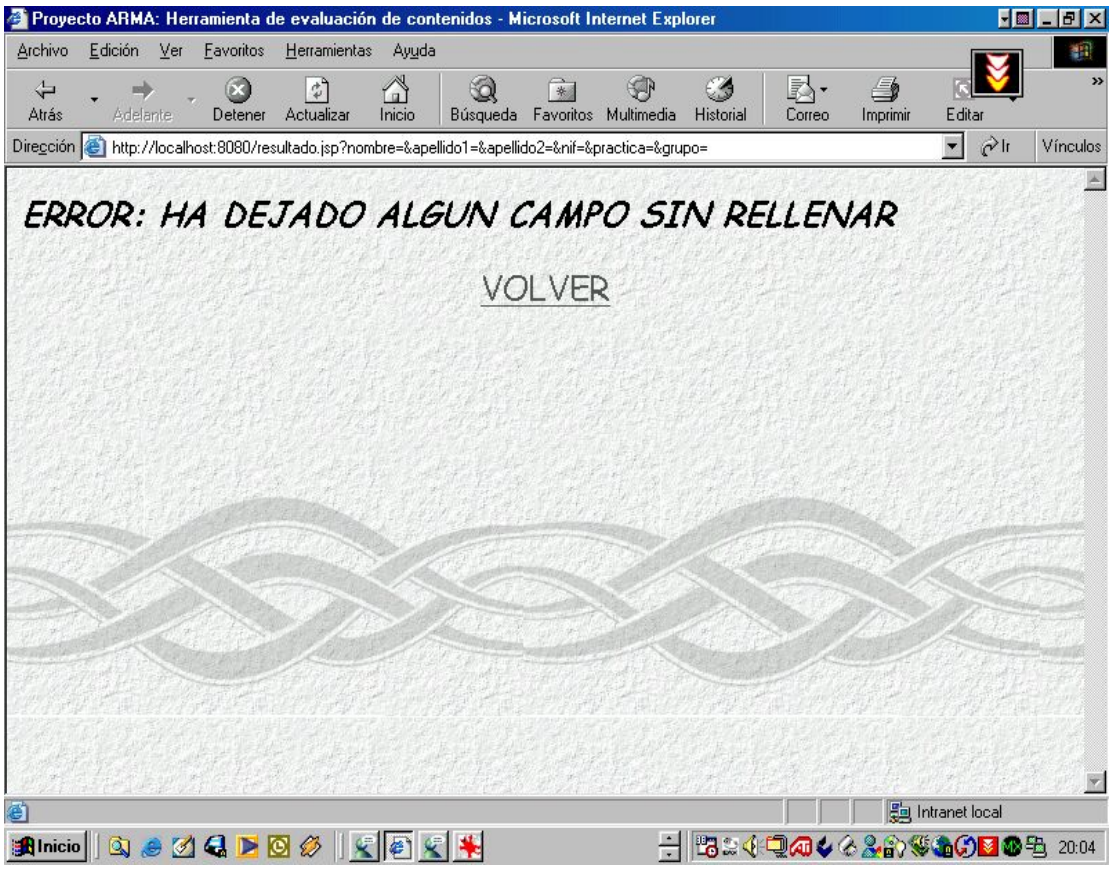
Aparece la siguiente pantalla



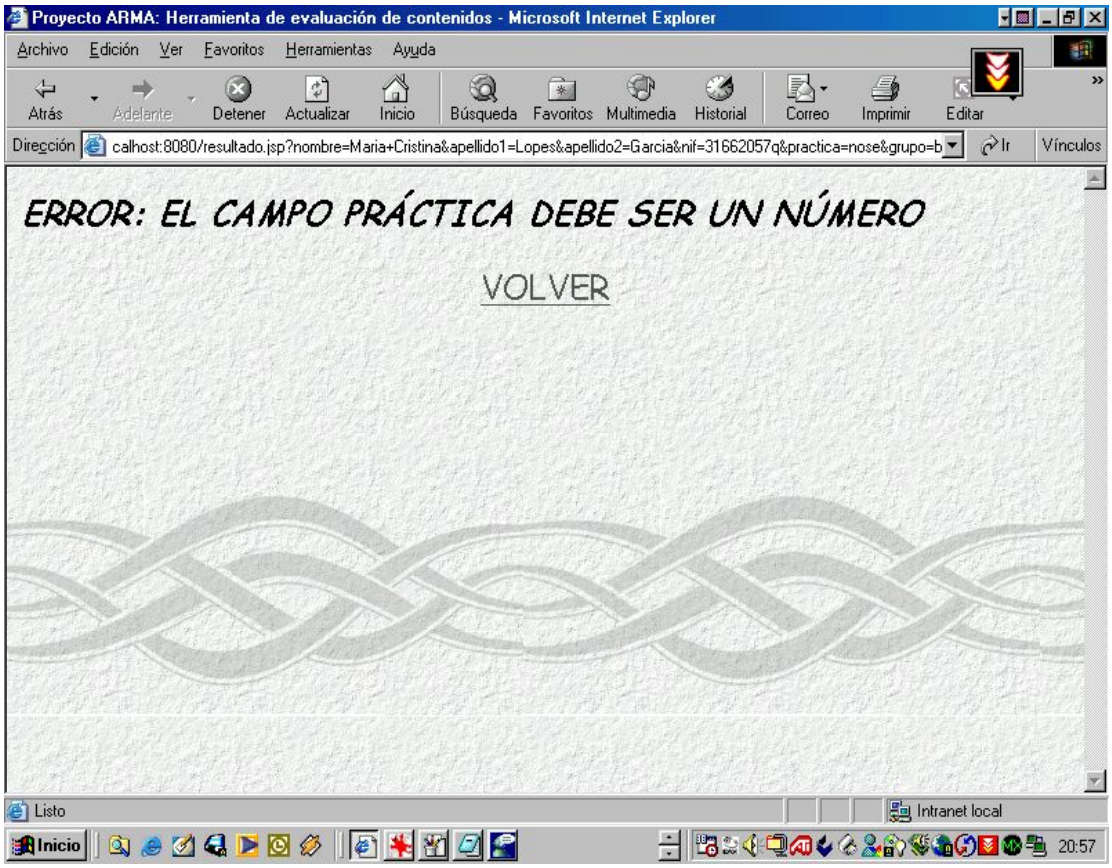

En la captura de programas el alumno debe introducir el nombre, los apellidos, el NIF, el grupo, y pulsar el botón siguiente.

Está habilitado un botón de borrar que resetea completamente el formulario. En caso de que alguno de los datos no sea correcto al pulsar el botón siguiente, aparecerá un mensaje de error indicando en cada caso el tipo de error:

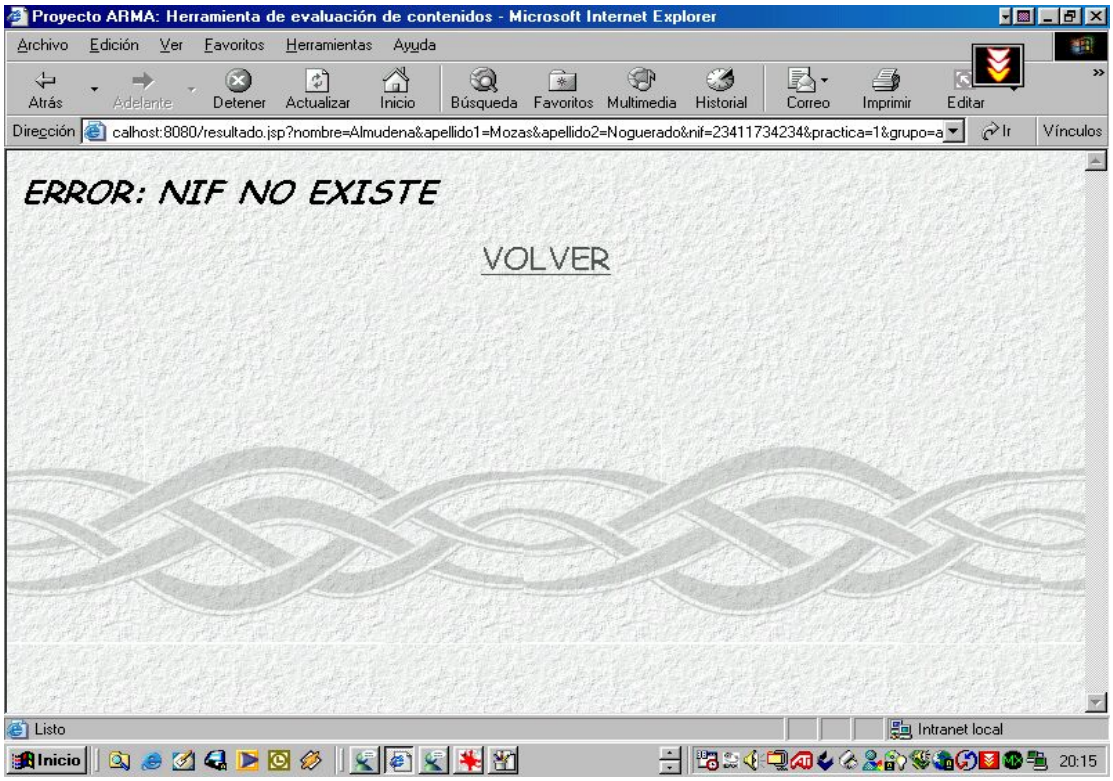
- En el caso de dejar algún campo sin rellenar:



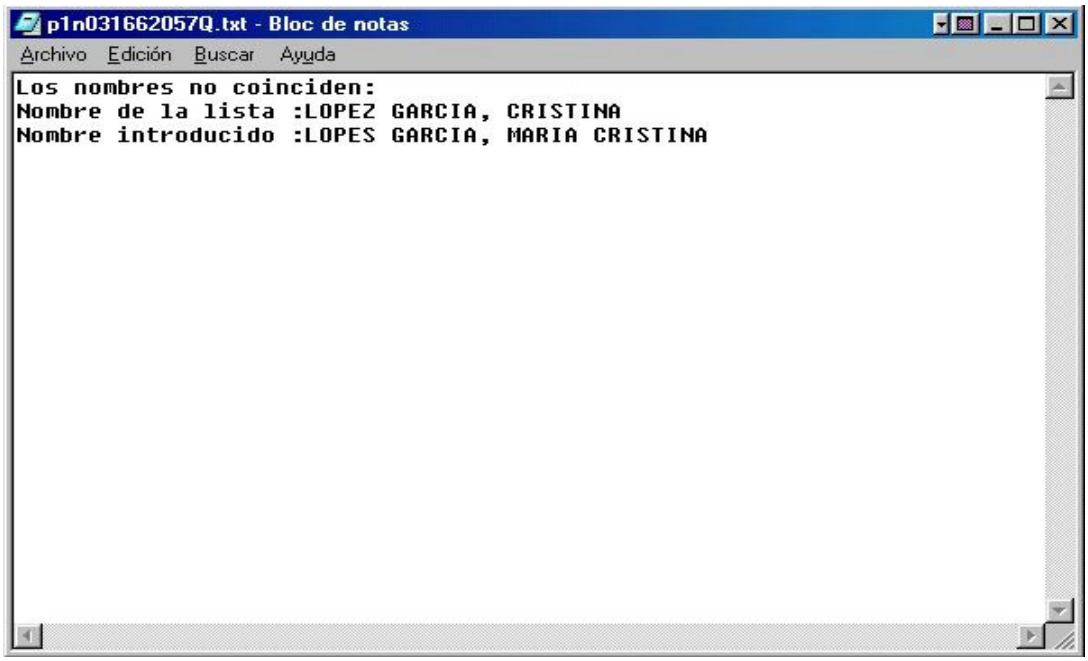
- En caso de que el formato del número de práctica sea incorrecto:



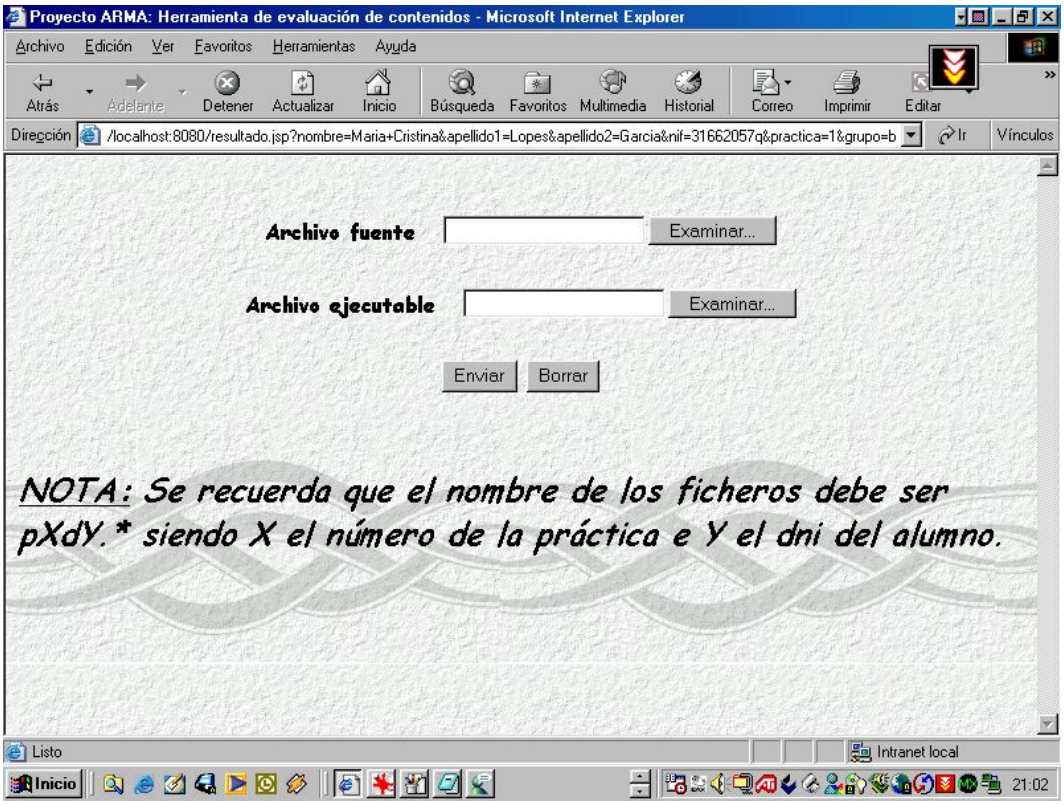

- En el caso de que el NIF no exista en la lista de alumnos:



En caso de que estén todos los datos rellenos, y el NIF sea correcto, pero el nombre del alumno introducido no coincida con el que tiene el profesor en su lista de alumnos se genera un archivo de warning en la carpeta: c:/jswdk1_0_1/webpages/upload/warnings/ como por ejemplo:

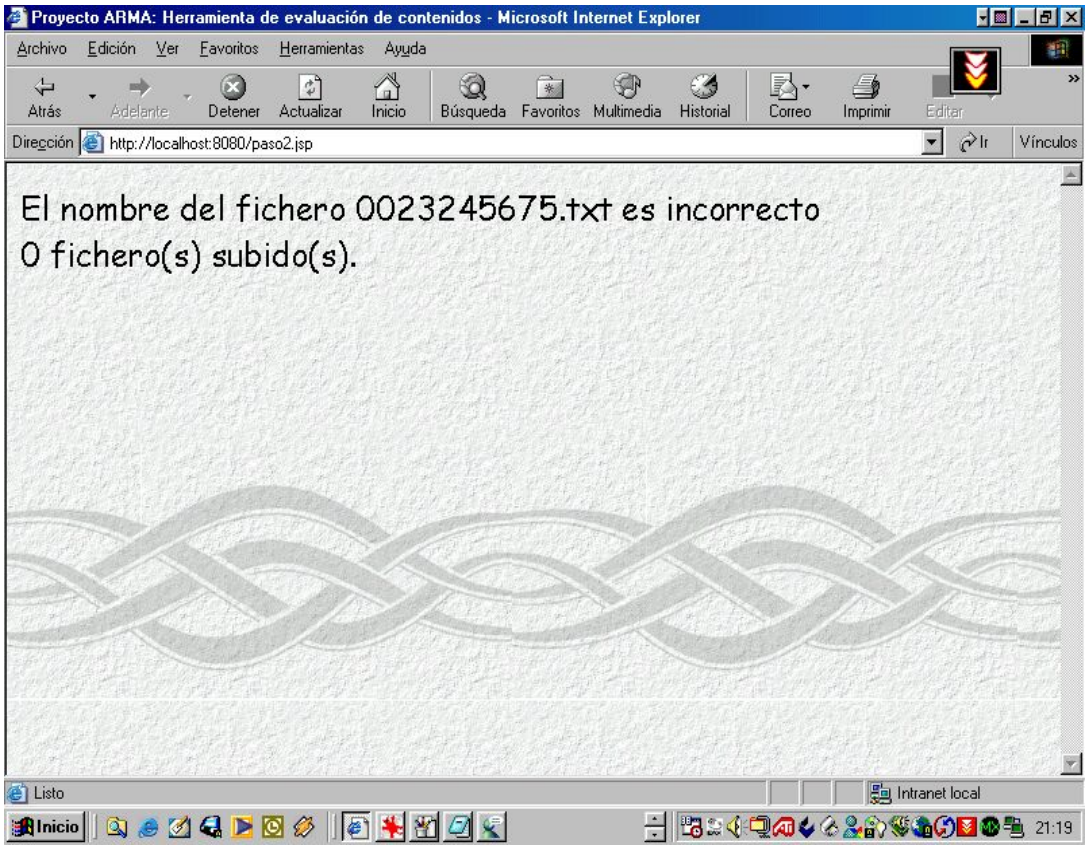



En caso de haber completado los datos correctamente se le permite al alumno incluir los archivos que desea mandar. Para ello hay habilitados unos cuadros de texto desde los que puede seleccionar los archivos a mandar, tanto los fuente como los ejecutables.



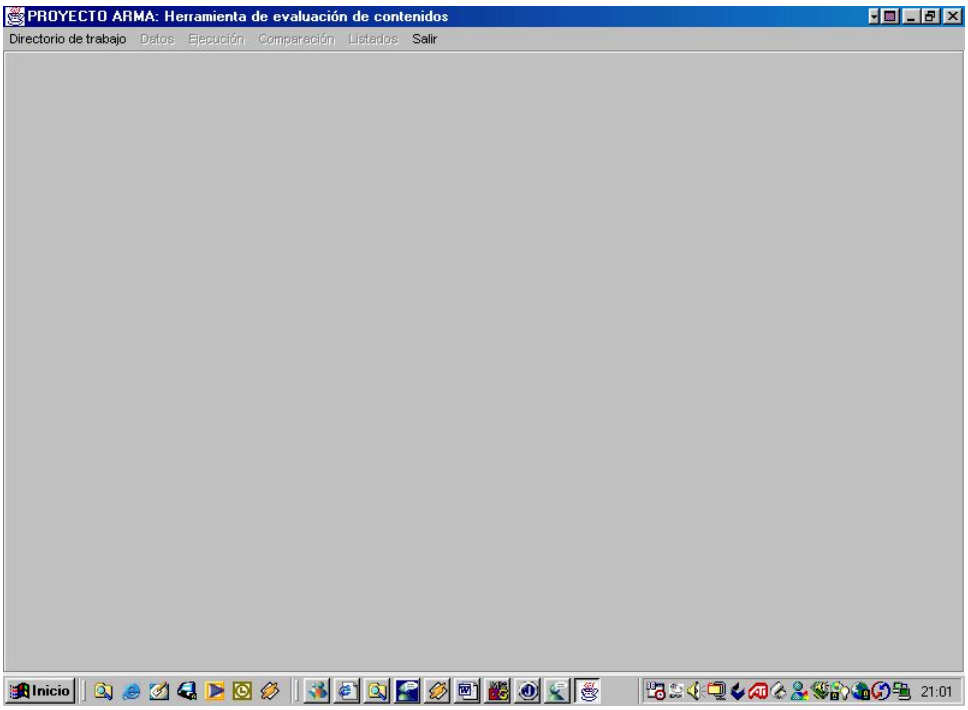
Una vez enviados los archivos, quedan almacenados en la carpeta practica*NºPrac* que ya estaba creada previamente.

En el caso de que las prácticas que intenta subir el alumno no respeten el formato deseado se informa de este hecho al alumno y no se suben las prácticas al servidor.

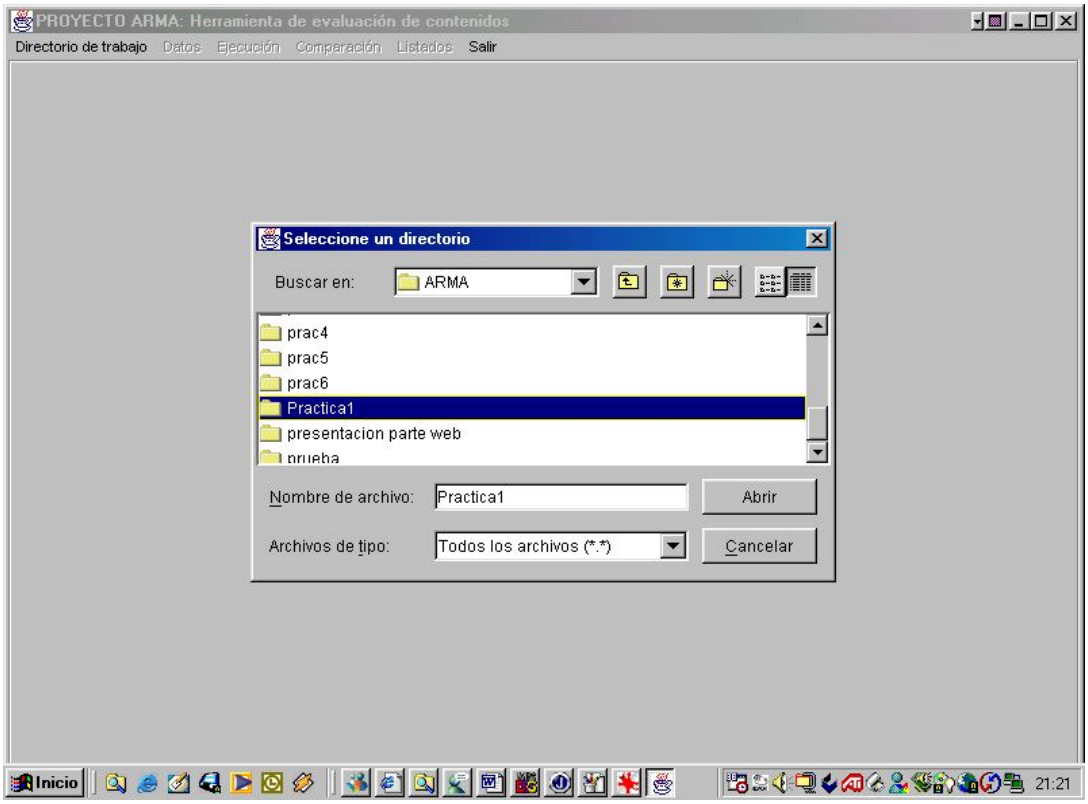


## 4.2.- HERRAMIENTA DE EVALUACIÓN DE CONTENIDOS

Al abrir la aplicación se muestra la siguiente pantalla.




donde se encuentran activas únicamente las opciones de salir del programa y de seleccionar el directorio de trabajo donde se encuentran las prácticas qué queremos corregir. El siguiente paso es seleccionar el directorio de trabajo, para ello seleccionamos la carpeta donde se encuentran las prácticas y pulsamos el botón abrir.



Tras este paso se nos activan el resto de las opciones del programa:

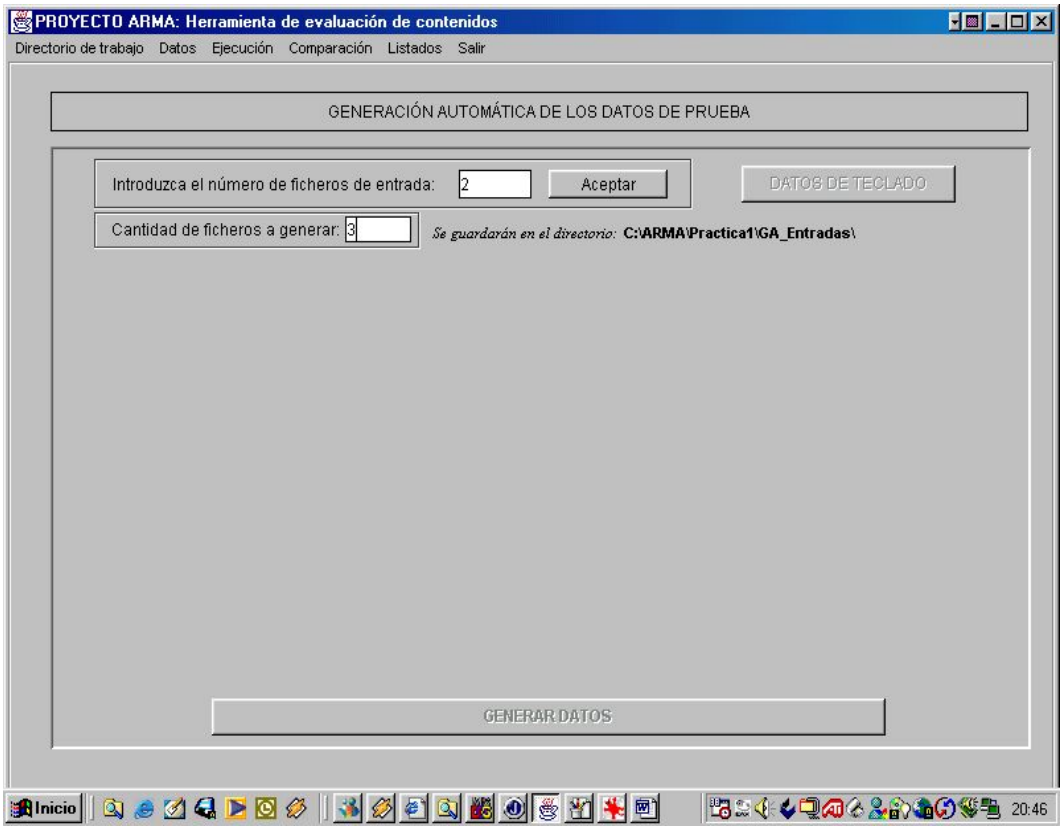
4.2.1.- Datos: generación de los datos de entrada de las prácticas, tenemos dos opciones:

4.2.1.1.- Generación automática: el programa genera ficheros de entrada con datos aleatorios.
4.2.1.2.- Generación manual: el programa selecciona ficheros de entrada ya existentes.

## 4.2.1.- Pantallas de generación de los datos

### 4.2.1.1.- Generación automática de los datos

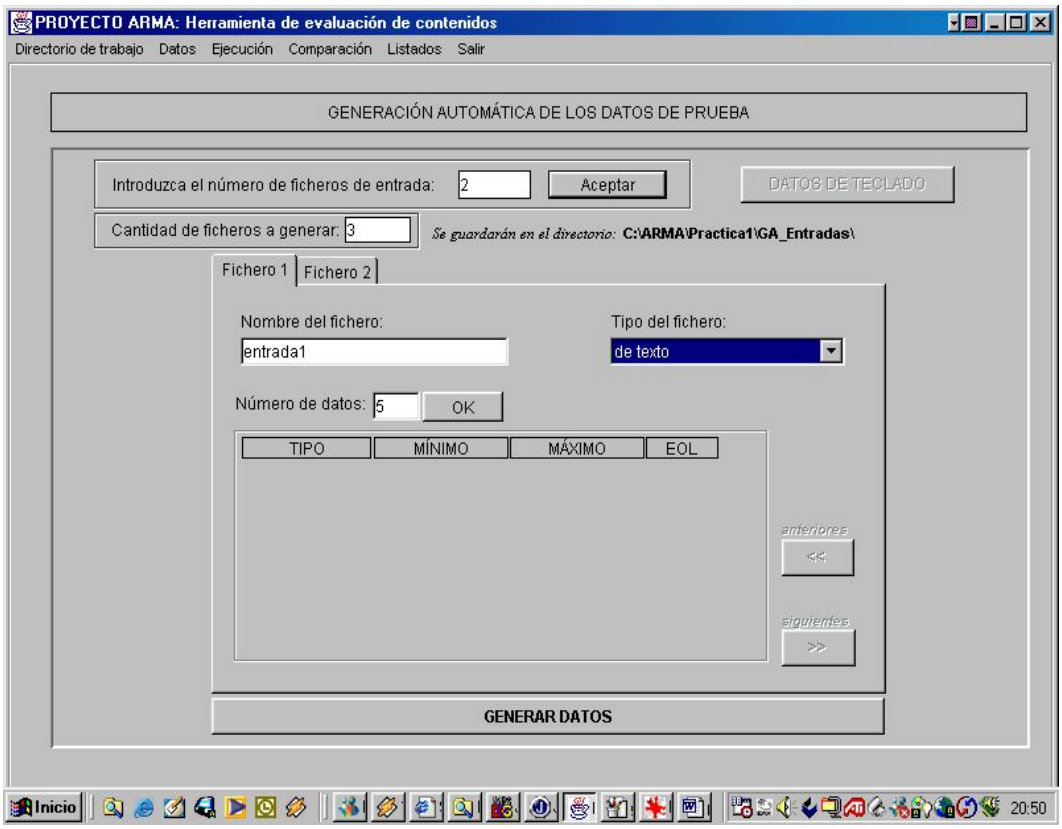
La pantalla inicial de la generación automática es la siguiente:




donde:

- *Número de ficheros de entrada*: es el número de ficheros de entrada que necesitan las prácticas para ser ejecutadas.
- *Datos de teclado:* se piden los datos de entrada por teclado que serán proporcionados al programa al ejecutarlo. Esta opción no se encuentra disponible en esta versión de la herramienta.
- *Cantidad de ficheros a generar*: número de ficheros con datos aleatorios distintos que se van a generar de cada fichero de entrada.

Una vez introducidos estos datos debemos rellenar el formato que queremos para cada uno de los ficheros de entrada:



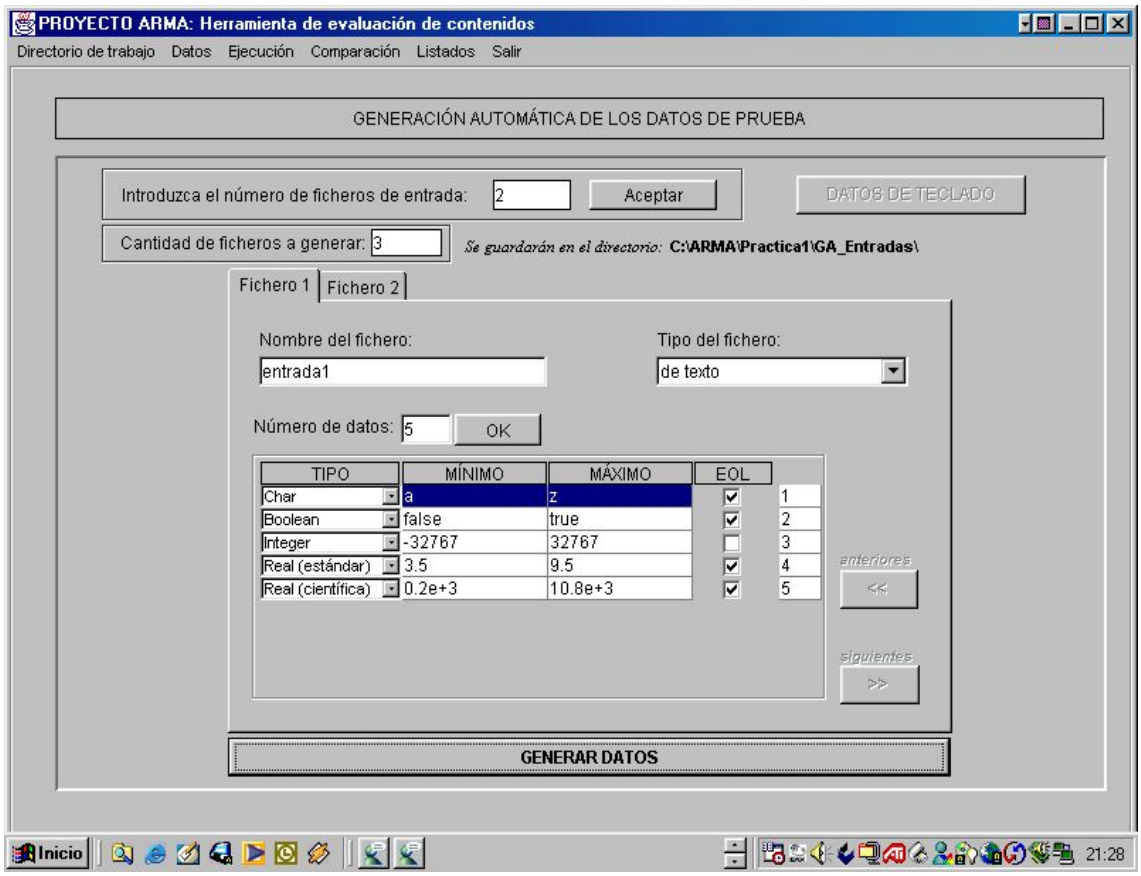

donde:

- *Nombre del fichero*: es el nombre del fichero de entrada que usan los alumnos en las prácticas (coincidirá en todas las prácticas realizadas).

- *Tipo de fichero*: puede ser de texto o binario. La opción de fichero binario no se encuentra disponible en esta versión de la herramienta.

- *Número de datos*: número de datos que va a haber en el fichero de entrada.

Tras introducir el número de datos que necesita nuestro fichero de entrada tenemos que introducir el tipo de cada uno de ellos y el rango entre el que estarán comprendidos.



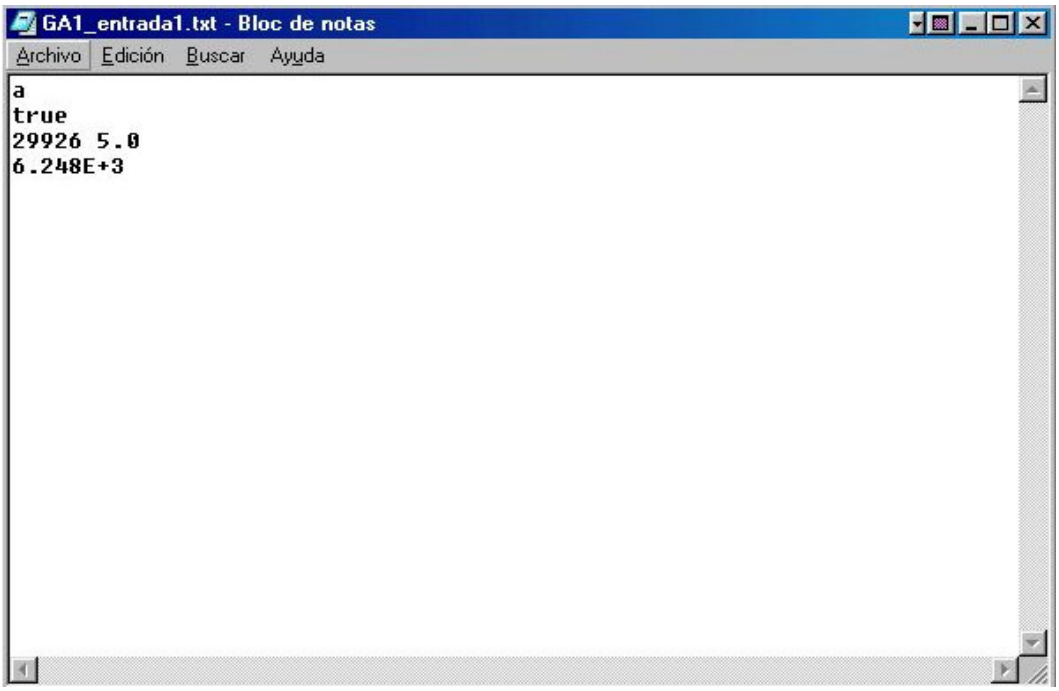

Podemos introducir datos de 4 tipos diferentes: char, boolean, integer y real (en formato estándar o científico). Para introducir los datos pinchamos en la casilla correspondiente y después de introducir cada dato pulsamos intro. También podemos indicar si a continuación de cada dato queremos introducir un fin de línea o no. El programa controla que no se haya dejado ningún campo en blanco, que el formato de los valores introducidos en los campos mínimo y máximo se correspondan con el formato correspondiente del campo tipo así como que el valor del mínimo sea menor que el del máximo. En caso de no cumplirse alguno de estos requisitos la herramienta informaría de ello con un mensaje.

Una vez introducidos todos los campos correctamente para cada uno de los ficheros de entrada y pulsado el botón generar datos, la herramienta generará tantos ficheros de entrada como le hayamos indicado, con unos datos aleatorios dentro de los límites que también le hemos indicado. Los ficheros se crean dentro del directorio de trabajo en la carpeta *GA_Entradas*. El nombre de cada uno de ellos seguirá el siguiente formato:

GA*NumGen_NomFichEntr*.txt

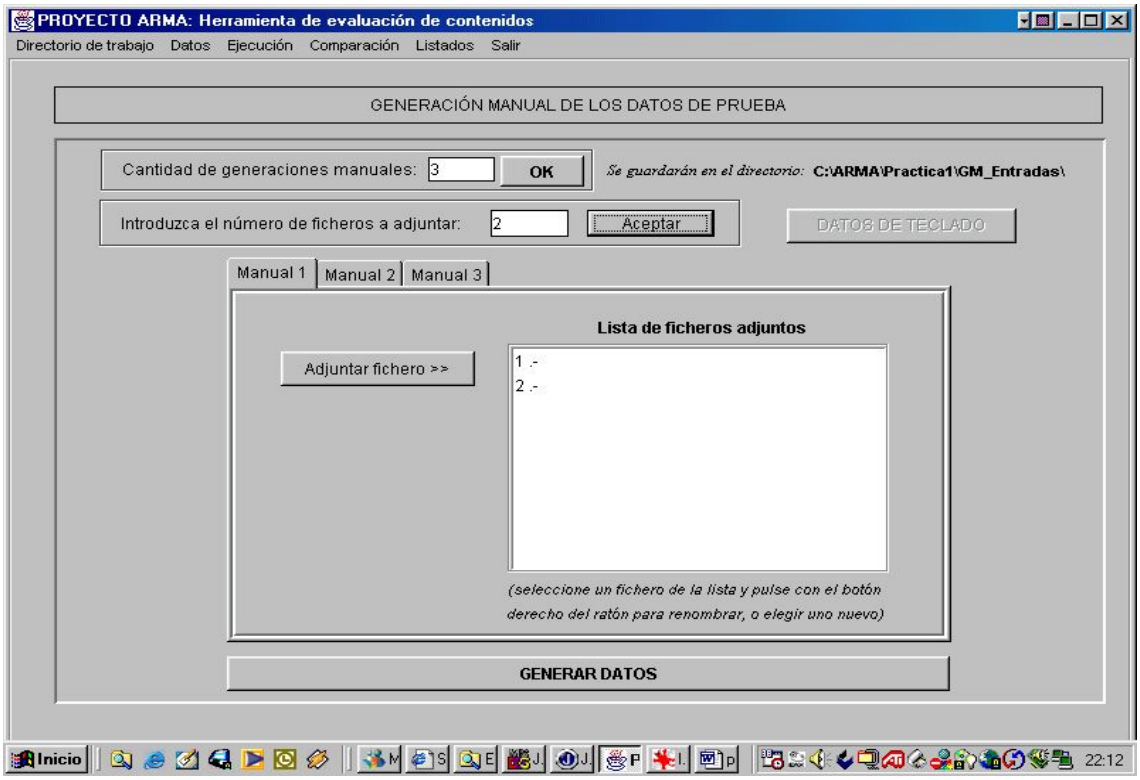
siendo *NumGen* el número de generación aleatoria a la que corresponde el fichero y *NomFichEn*t el nombre del fichero de entrada.

Un ejemplo de fichero generado sería el siguiente:




## 4.2.1.2.- Generación manual de los datos

Además de la posibilidad de generar datos de entrada de forma automática y por tanto con unos datos aleatorios, el profesor puede querer probar las prácticas de los alumnos con unos datos de entrada proporcionados por él y que se encuentren en unos ficheros determinados.



- *Cantidad de generaciones manuales*: número de generaciones manuales distintas que se quiere realizar.
- *Introduzca el número de ficheros a adjuntar*: número de ficheros de entrada que necesitan las prácticas para ser ejecutadas.

Para adjuntar cada fichero se debe pulsar el botón adjuntar fichero y seleccionar el fichero deseado. Se dará la opción de renombrarlos para el caso en que el nombre con el que se tienen guardados no coincida con el que debe emplearse en la práctica. Para renombrar un fichero se debe pulsar sobre él, botón derecho del ratón, seleccionar renombrar el fichero, darle el nuevo nombre y pulsar el botón aceptar. Una vez se hayan adjuntado y renombrado (si es necesario) todos los ficheros se pulsará el botón generar datos.

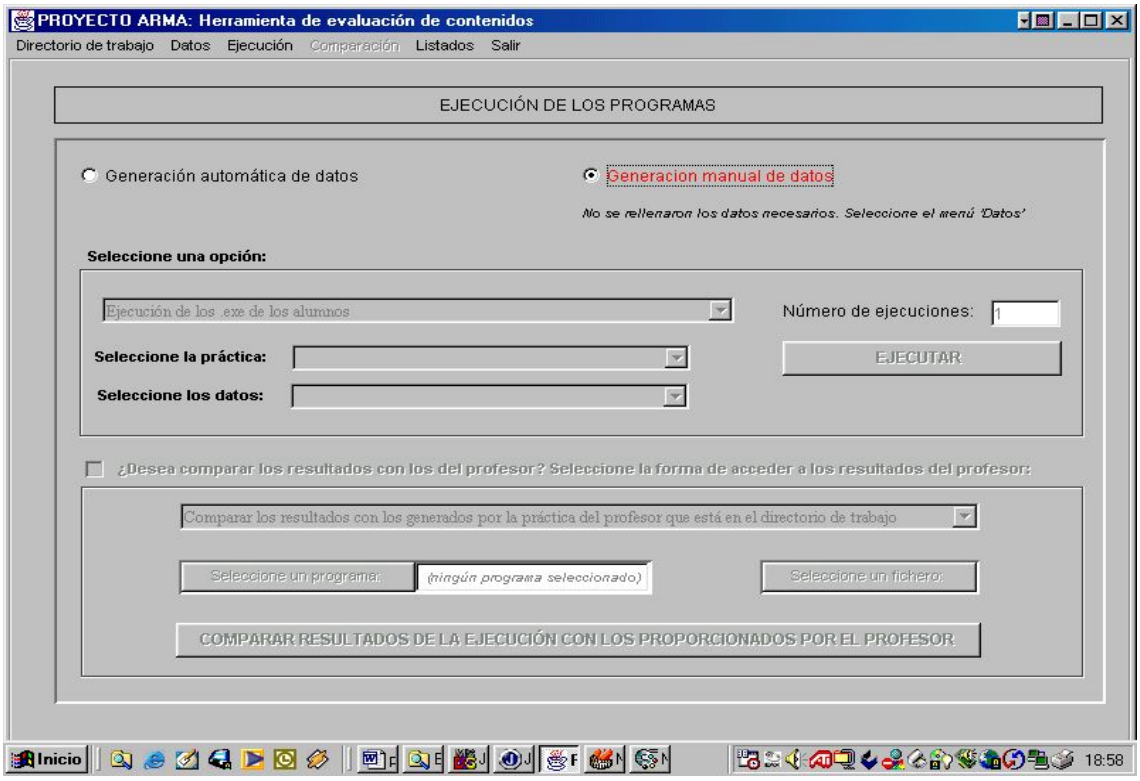
La herramienta controlará si falta algún fichero por adjuntar informando de ello con un mensaje si esta circunstancia se produce.

Tras pulsar el botón aceptar, la herramienta copia dichos ficheros en el directorio de trabajo en la carpeta /GM_Entradas/ y para cada una de las generaciones manuales que se han hecho crea una carpeta /GM_Entrada*NumGen*/ donde hace la copia de los ficheros seleccionados.

## 4.2.2.- Pantalla de ejecución de los programas de los alumnos

En esta pantalla es donde la herramienta compila y ejecuta las prácticas de los alumnos que se encuentran en el directorio de trabajo escogido. Para ejecutar las prácticas se debe seleccionar si se quieren usar los datos de entrada automáticos o los datos de entrada manuales que se han generado previamente. Si no se hubiera generado ninguno de ellos o se selecciona una opción que previamente no ha sido generada se presenta un mensaje de error y no se permite realizar la ejecución. En esta pantalla también se realizan las comparaciones de los resultados de la ejecución con los resultados del profesor.

### 4.2.2.1.- Ejecuciones

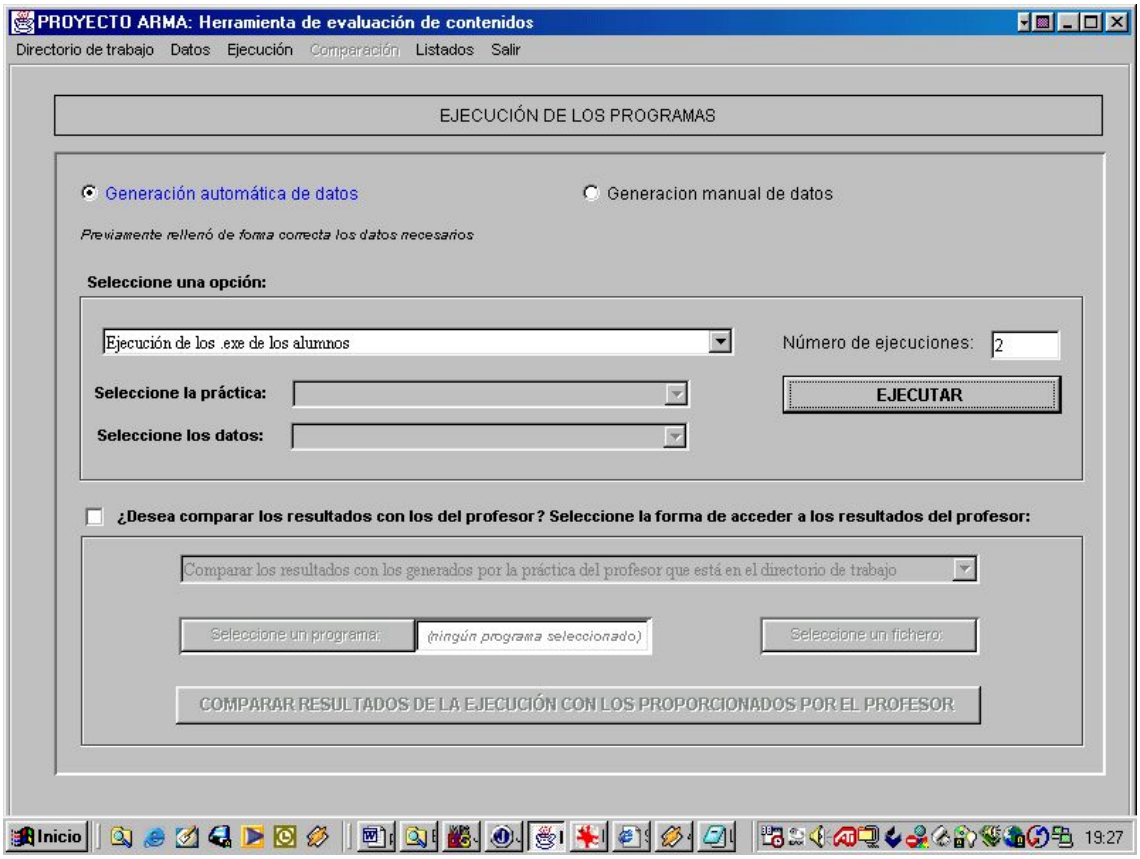


Una vez se ha seleccionado correctamente que datos de entrada se quieren utilizar se activa el recuadro correspondiente a la ejecución. Ésta presenta cuatro posibilidades diferentes:

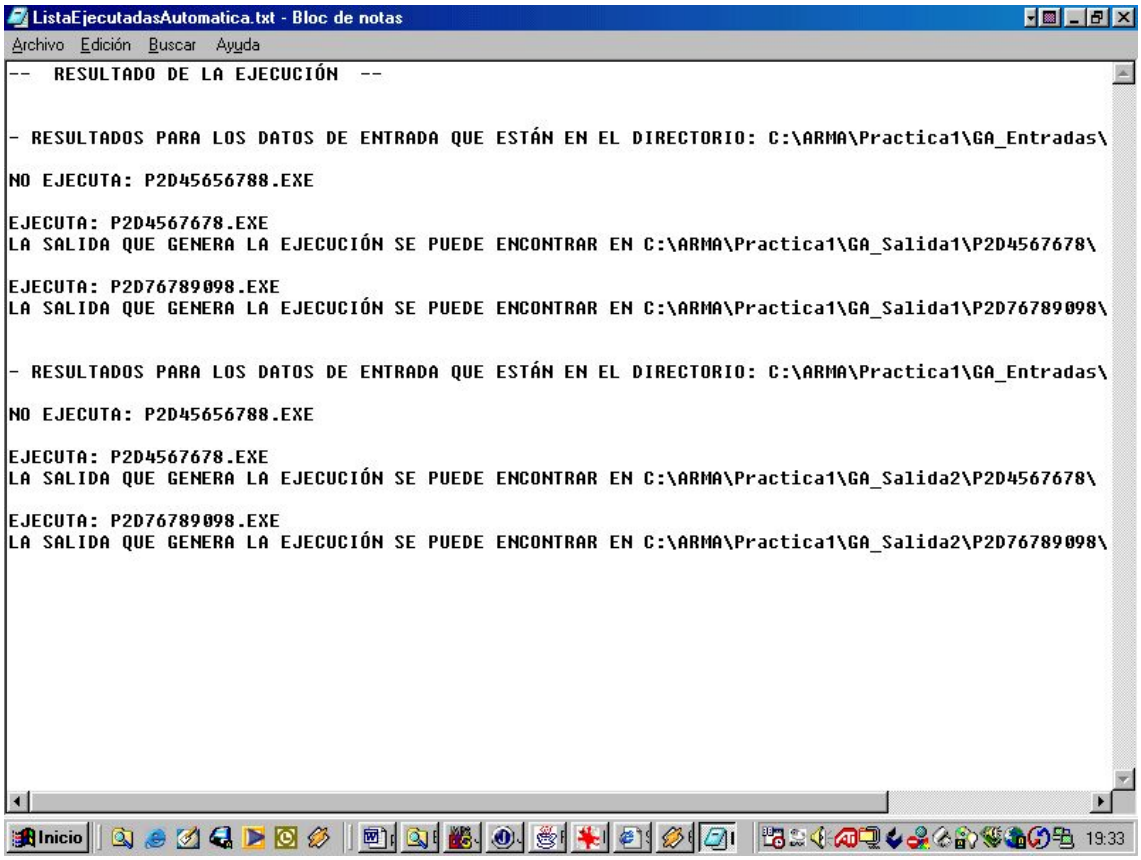
4.2.2.1.a.- Ejecución de los *.exe* de todos los alumnos.
4.2.2.1.b.- Ejecución de la práctica de un único alumno con unos datos de entrada específicos.
4.2.2.1.c.- Compilación y ejecución de todos los programas.
4.2.2.1.d.- Ejecución del *.exe* y comparación con la compilación y la posterior ejecución de dicha práctica.

### ***4.2.2.1.a.- Ejecución de los .exe de los alumnos***

Se ejecutarán todos los *.exe* de los alumnos que se encuentren en el directorio de trabajo. Primero se debe seleccionar en el recuadro de Número de ejecuciones el número de veces que se quiere probar el programa. Lo normal es que este número coincida con el número de archivos que previamente hemos generado. Si se selecciona un número menor o mayor se mostrara un mensaje de aviso. Una vez hecho esto se pulsa el botón ejecutar. Para cada conjunto de datos de entrada se ejecutarán todos los *.exe* que haya en el directorio de trabajo.




Tras realizarse la ejecución de todas las prácticas se crea en el directorio de trabajo un fichero de texto de nombre ListaEjecutadas*TipoGen*.txt siendo TipoGen Automatica o Manual dependiendo de la generación elegida donde se anota si ejecuta o no ejecuta cada práctica, donde se encuentran los datos de entrada con los que se ha ejecutado y donde se encuentran las salidas que la ejecución haya generado.

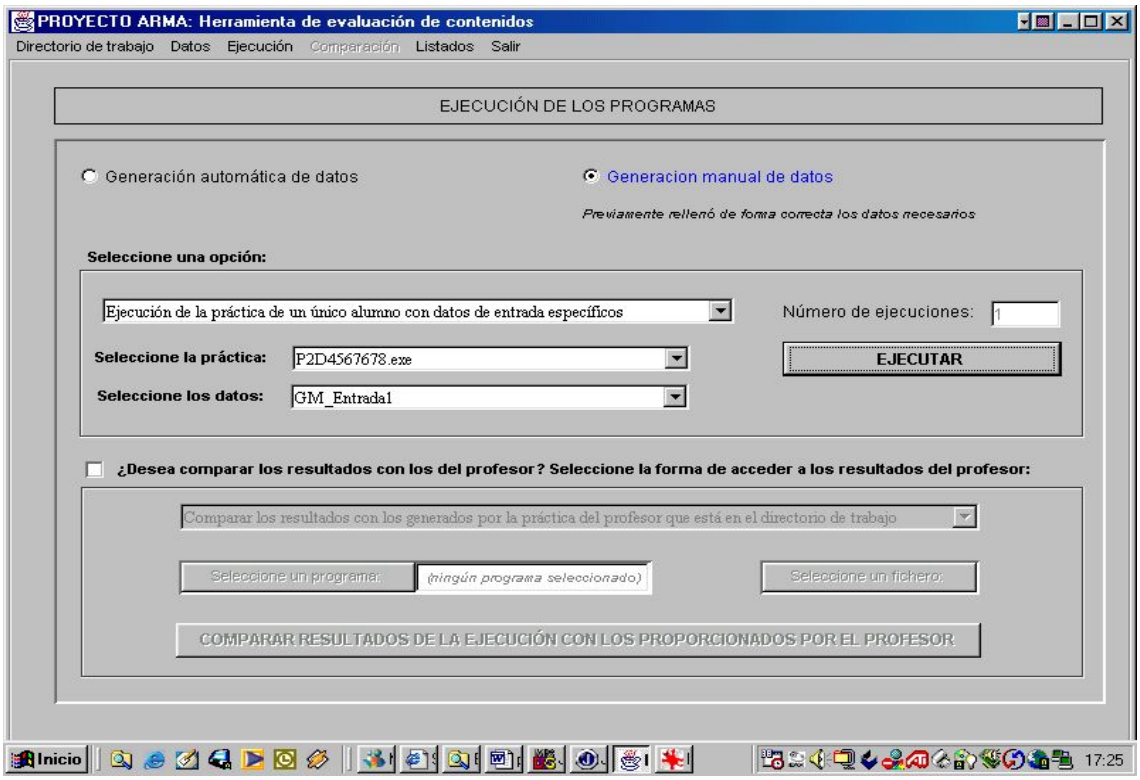



Para cada conjunto de datos de entrada para el que se hayan ejecutado las prácticas se crea un directorio en el directorio de trabajo de nombre G*TipoGen*_Salida*Num* siendo *TipoGen* A o M dependiendo de si la ejecución utilizó los datos de la generación automática o de la generación manual y *Num* el número de conjunto de datos de entrada utilizado.

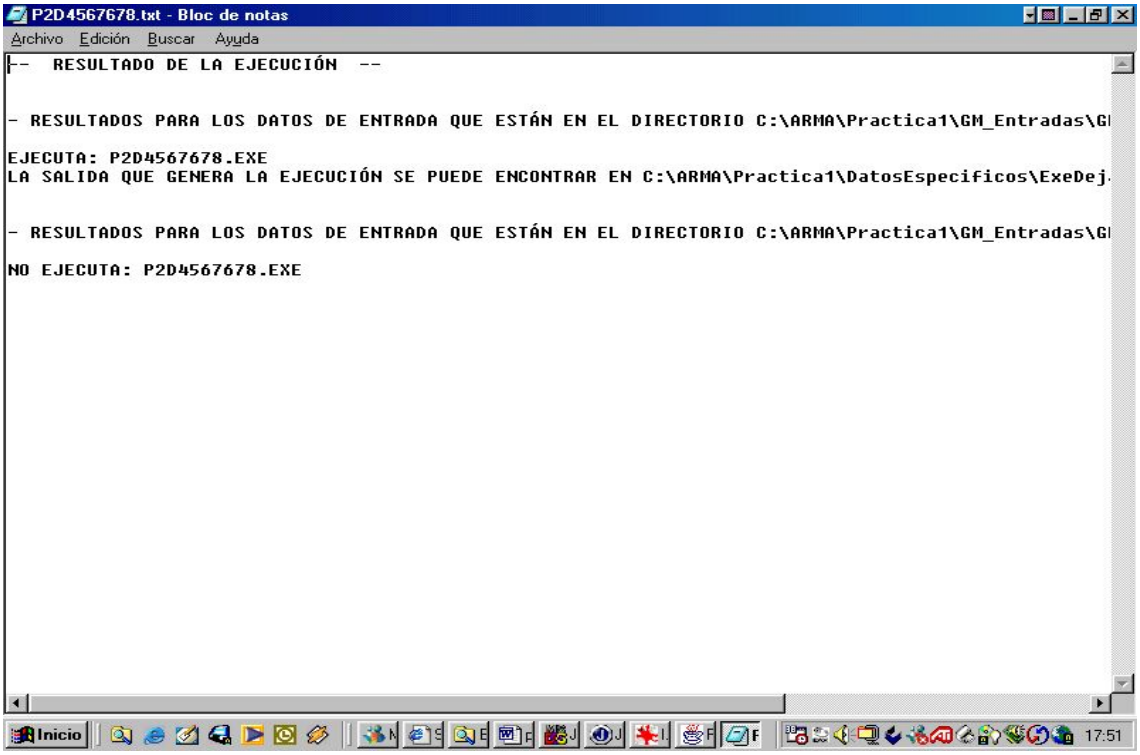
Dentro de éste directorio se crea una carpeta para cada alumno con los nombres de las prácticas y en cada una de ellas se encontrarán las salidas que la ejecución de la práctica de ese alumno haya generado.

#### ***4.2.2.1.b.- Ejecución de la práctica de un único alumno con unos datos de entrada específicos***

Esta opción sólo se encuentra disponible para la generación manual. Una vez seleccionada esta opción se debe seleccionar la práctica que se desea ejecutar y los datos de entrada con los que se desea hacerlo. Los ejecutables que se pueden seleccionar son los que han dejado los alumnos o los que se han producido al compilar las prácticas (si se hubiera hecho previamente). El número de ejecuciones es uno y no se puede cambiar. Tras esto pulsamos el botón ejecutar.



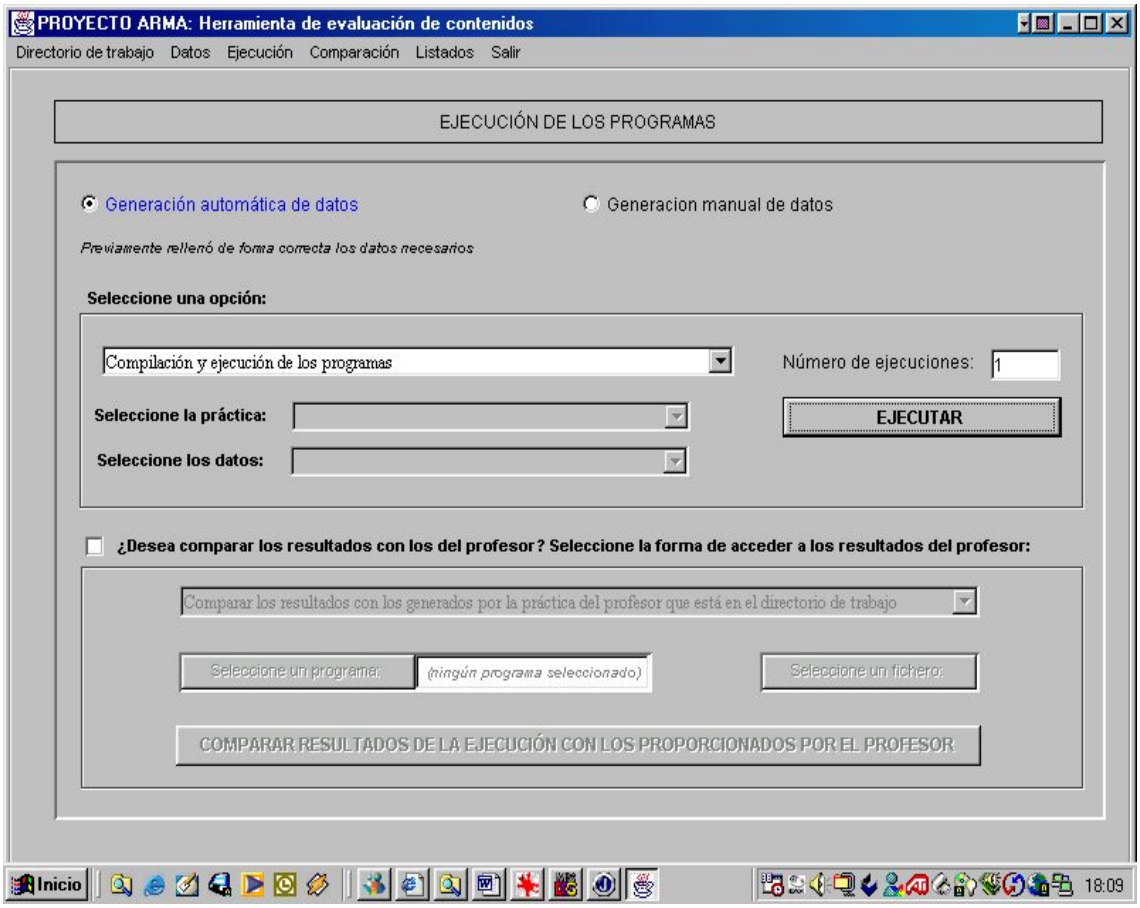

El resultado es que se crea una carpeta llamada DatosEspecificos para almacenar los resultados de este tipo de ejecución. Dentro de ésta se creará una carpeta ExeGenerados si la práctica que se ha ejecutado era un exe generado tras la compilación o ExeDejados si la práctica que se ha ejecutado era una dejada por el alumno. Dentro de cualquiera de las dos carpetas (dependiendo del caso que se haya ejecutado) se creará una carpeta con el nombre de la práctica sin la extensión. Aquí encontraremos un archivo de texto de nombre NombrePractica.txt con la información sobre las distintas pruebas con diferentes datos específicos que se hayan hecho con esta práctica.



También en la carpeta donde se encuentra este archivo encontraremos una carpeta con las salidas que generará cada ejecución que haya tenido éxito.

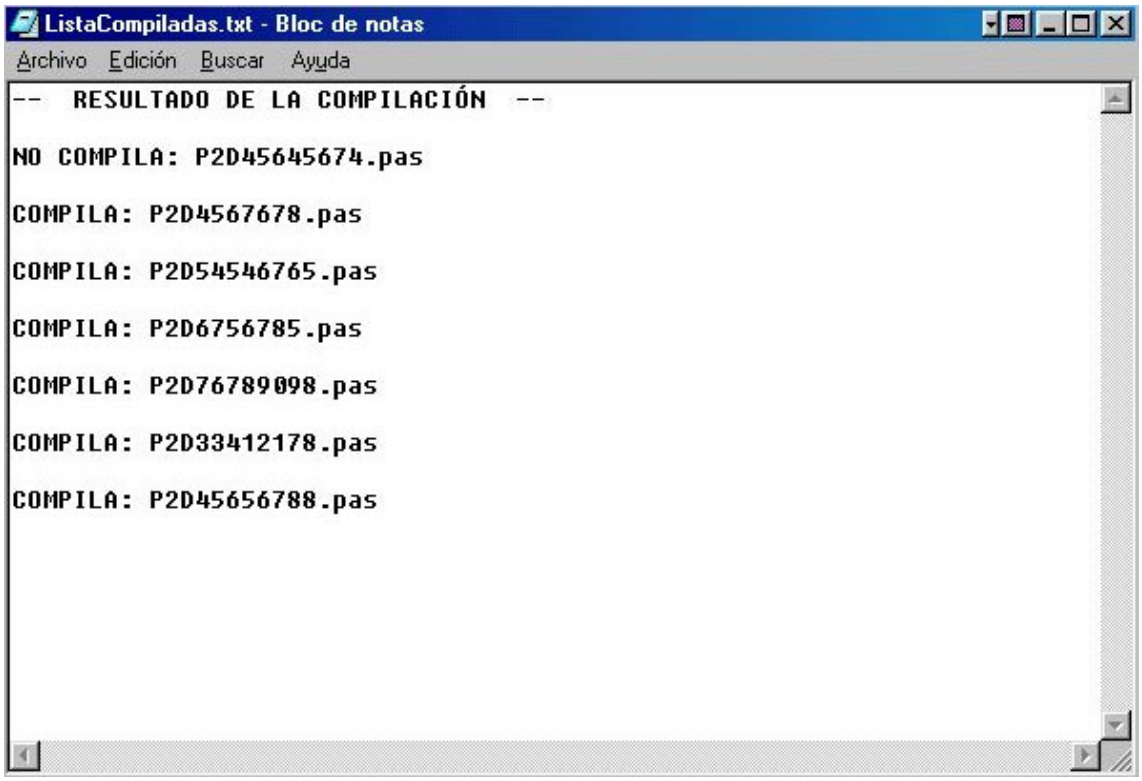
### *4.2.2.1.c.- Compilación y ejecución de los programas*

Esta opción primero compila los programas *.pas* que se encuentran en el directorio de trabajo, generando sus ejecutables correspondientes y ejecutando después estos. Simplemente debemos seleccionar el número de ejecuciones que deseamos realizar a los *.exe* generados tras la compilación y pulsar el botón ejecutar.




Se crea una carpeta Compiladas donde se van a almacenar todos los resultados de esta opción.

Se crea un fichero de texto de nombre ListaCompiladas.txt donde indicamos si cada práctica del directorio de trabajo compila o no.

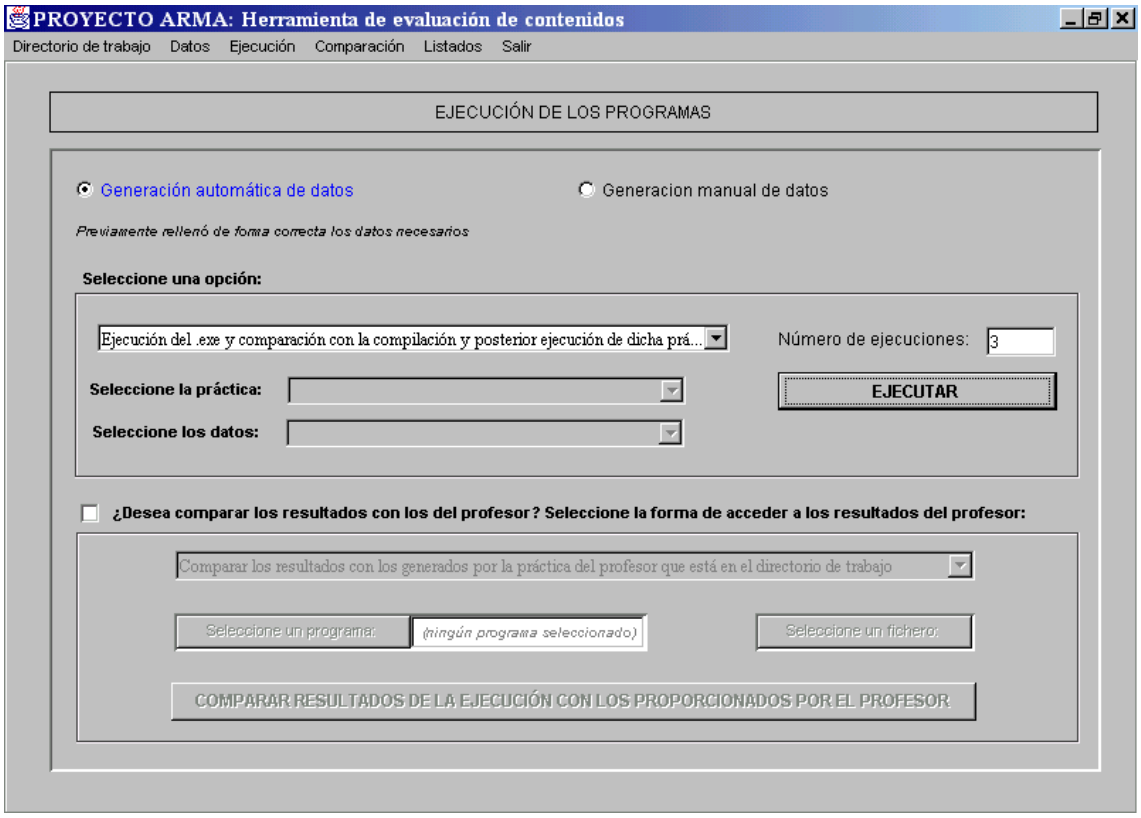



Dentro de la carpeta Compiladas se crea una carpeta ExeGenerados donde tendremos los ejecutables que se han generado al compilar las prácticas. Además de estos tendremos una nueva lista llamada ListaEjecutadas*TipoGen*.txt donde encontramos la información sobre si estos ejecutables generados ejecutan o no, con qué datos se ha probado y donde se encuentra la salida que generan en el caso de que sí ejecuten.

Para cada conjunto de datos de entrada para el que se hayan ejecutado las prácticas se crea un directorio dentro de ExeGenerados de nombre G*TipoGen*_Salida*Num* siendo *TipoGen* (GA) o (GM) dependiendo de si la ejecución utilizó los datos de la generación automática o de la generación manual y Num el número de conjunto de datos de entrada utilizado. Dentro de éste directorio se crea una carpeta para cada alumno cuyos nombres de las prácticas y en cada una de ellas se encontrarán las salidas que la ejecución de la práctica de ese alumno haya generado.

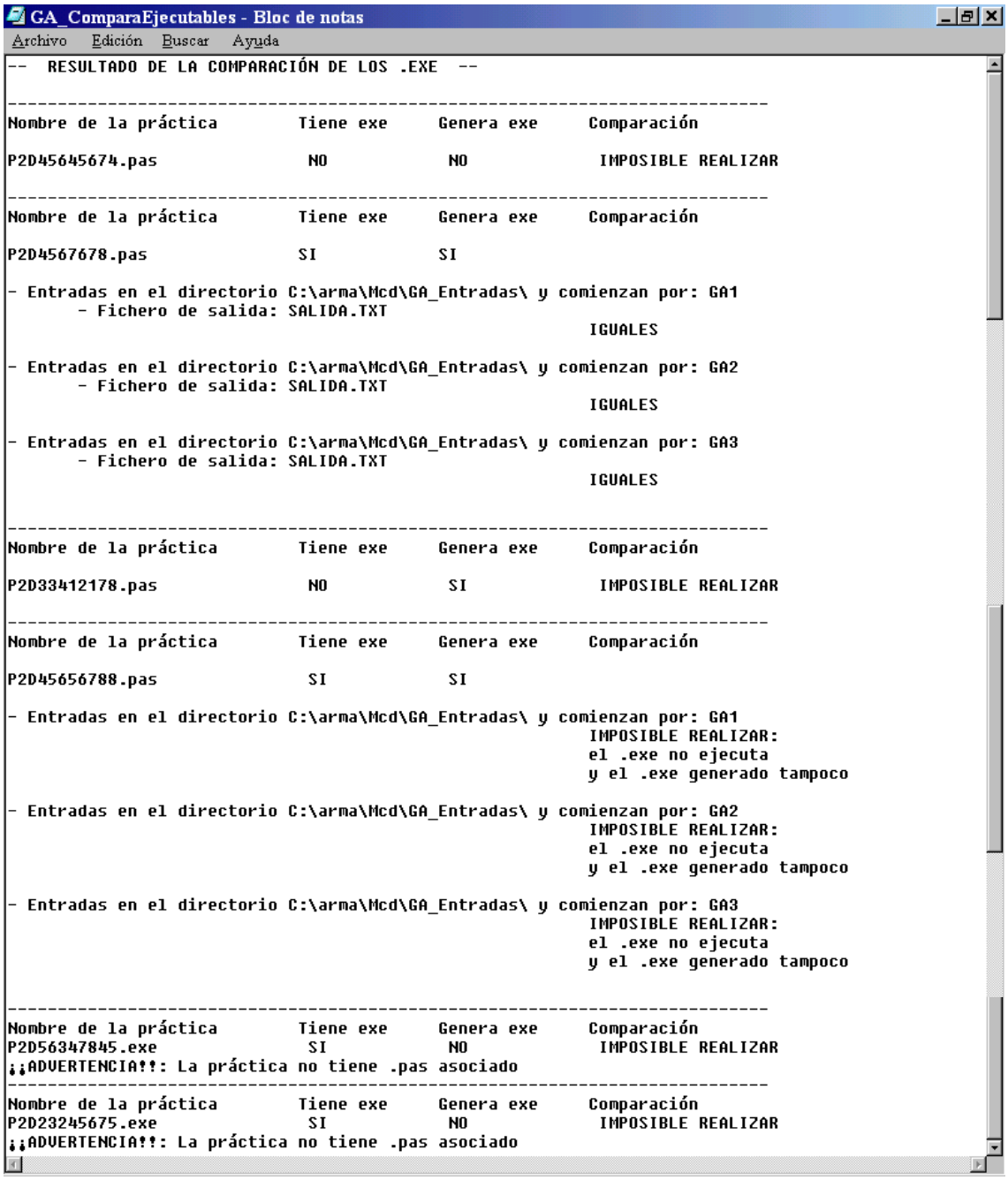
#### ***4.2.2.1.d.- Ejecución del .exe y comparación con la compilación y la posterior ejecución de dicha práctica***

Esta opción primero ejecuta los programas *.exe* que se encuentran en el directorio de trabajo; posteriormente compila los programas *.pas* que se encuentran en el directorio de trabajo generando sus ejecutables correspondientes y ejecutando después estos. Después se realiza la comparación entre los resultados obtenidos de ambas ejecuciones. Simplemente debemos seleccionar el número de ejecuciones que deseamos realizar y pulsar el botón ejecutar.




Esta opción supone realizar de forma secuencial y automática los tipos de ejecuciones uno y tres explicadas anteriormente (puntos 4.2.2.1.a.- y 4.2.2.1.c.- respectivamente), y además añade la funcionalidad de generar la información relativa a la comparación de ambas ejecuciones. De esta manera, la información generada para las ejecuciones es la misma que la explicada para los puntos de *'ejecución de los .exe de los alumnos'* y de *'compilación y ejecución de los programas'*. Como característica nueva, se añade en el directorio de trabajo un fichero de texto de nombre *TipoGen*_ComparaEjecutables.txt siendo *TipoGen* Automatica (***GA***) o Manual (***GM***) dependiendo de la generación elegida. En este archivos se anota para cada programa (*.exe*, o *.pas*) que se encuentra en el directorio de trabajo, el resultado de intentar realizar la comparación entre los ficheros de salida generados por el ejecutable dejado por el alumno y los ficheros de salida generados por el ejecutable generado como resultado de compilar el *.pas* asociado del alumno, para cada una de las entradas para las que se realizaron las ejecuciones. Estos ficheros de salida, pueden ser:

- *IGUALES*,
- *DISTINTOS*,
- o puede que sea *IMPOSIBLE REALIZAR* la comparación por diferentes motivos:
    - ó bien que el alumno no haya dejado el ejecutable
    - ó bien que el alumno no haya dejado el *.pas*, ó que habiendo dejado el *.pas*, la práctica no compile (en ambos casos es imposible generar el ejecutable)
    - ó bien que teniendo tanto el *.exe* como el *.exe* generado, alguno de los, o los dos programas no ejecuten.
    - ó bien que los nombres de las salidas (para el ejecutable y para el ejecutable generado) no coincidan de manera que sea imposible identificarlas, y por lo tanto imposible realizar la comparación.




Tras realizar alguna o varias de las ejecuciones anteriores *se puede comparar el resultado de la ejecución con el resultado de la ejecución de la práctica del profesor*.

## 4.2.2.2.- Comparación de los resultados

*La comparación* de los resultados *se realiza siempre sobre la última ejecución realizada*, de manera que así se pueden ir comprobando todos los resultados que se deseen, seleccionando la opción correspondiente después de cada ejecución.

Cuando se realiza esta función se crea en el directorio de trabajo una carpeta llamada 'CompConProf' que guardará toda la información relativa a las diferentes comparaciones que se vayan haciendo después de efectuar algún tipo de ejecución.

La comparación puede encontrar que los ficheros de salida sean:

- *IGUALES*,
- *DISTINTOS*,
- o puede que sea *IMPOSIBLE REALIZAR* la comparación por diferentes motivos.

También se anota información que puede ser relevante para el usuario, como por ejemplo cuando es imposible realizar una comparación y los motivos por los que no se ha podido llevar a cabo. Además, se muestran advertencias como por ejemplo en el caso de que se realice una comparación con algún ejecutable que no tiene asociado un programa *.pas*, para que el usuario sea capaz de valorar los resultados de la comparación.

La aplicación proporciona diferentes formas de acceder a los resultados del profesor:

4.2.2.2.a.- Comparar los resultados con los generados por la práctica del profesor:

4.2.2.2.b.- Ejecutar un programa para obtener los resultados

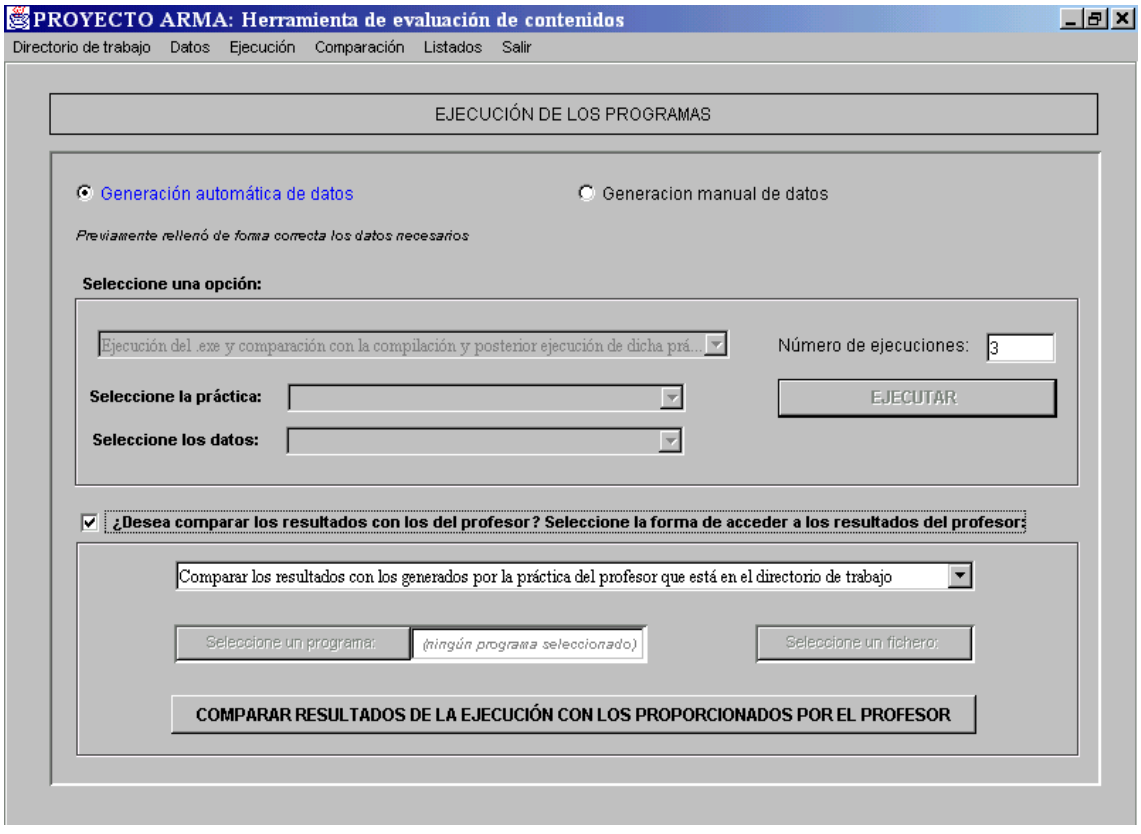
4.2.2.2.c.- Los resultados se obtendrán de un fichero

El usuario simplemente debe seleccionar alguna de estas opciones y pulsar el botón *COMPARAR RESULTADOS DE LA EJECUCIÓN CON LOS PROPORCIONADOS POR EL PROFESOR.*

Las distintas formas de acceder a los resultados del profesor son:

***4.2.2.2.a.- Comparar los resultados con los generados por la práctica del profesor***

Como se ha dicho anteriormente, la práctica del profesor debe encontrarse en el directorio de trabajo con el formato de nombre estándar siguiente: **P*x*PROF**, siendo ***x*** el número de práctica correspondiente. Esta comparación se realiza por tanto con los resultados de la ejecución de dicha práctica del profesor para las ejecuciones que se hubieran realizado con los tipos de datos específicos también seleccionados.



En la nueva carpeta, 'CompConProf', creada dentro del directorio de trabajo, se crean ficheros de texto con diferentes nombres, dependiendo del tipo de ejecución que se realizó:

- *tipoGen*_Ejecutables.txt (si la última ejecución realizada fue *'ejecución de los .exe de los alumnos'*)

- *tipoGen*_ExeGenerados.txt (si la última ejecución realizada fue *'compilación y ejecución de los programas'*)

- ambos ficheros de texto (*tipoGen*_Eejecutables.txt y *tipoGen*_ExeGenerados.txt), si la última ejecución realizada fue *'ejecución del .exe y comparación con la compilación y posterior ejecución de dicha práctica'*.

- *S*i la última ejecución realizada fue *'ejecución de la práctica de un único alumno con unos datos de entrada específicos'*, no se genera un listado genérico, sino un fichero de texto para la práctica concreta ejecutada con los datos de entrada específicos. En este caso en la carpeta 'CompConProf', se genera una subcarpeta llamada *DatosEspecificos* (si no estaba creada previamente) que contendrá los ficheros de texto e información relacionada con la comparación.

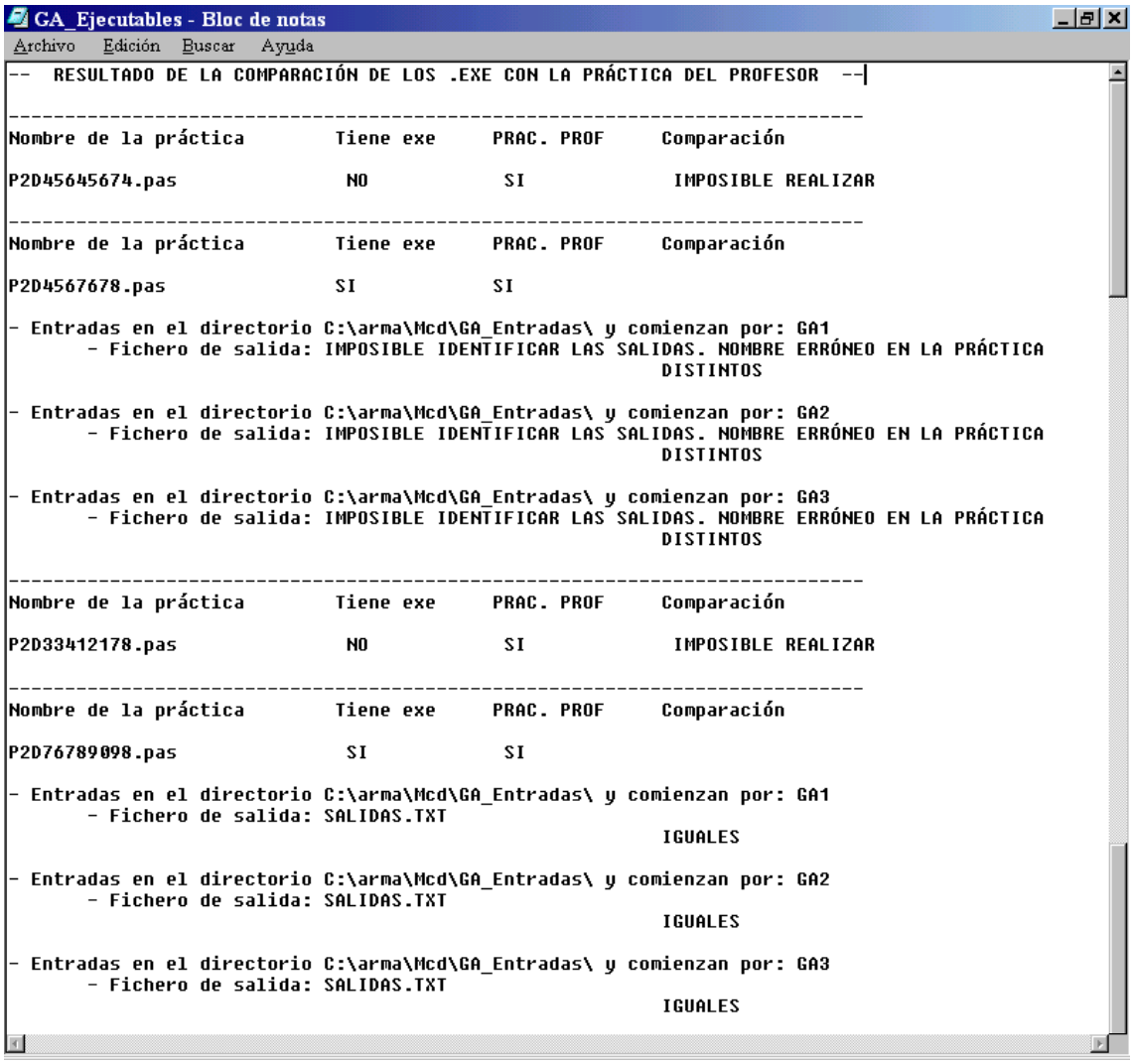
  En esta subcarpeta se guardan los resultados de la ejecución de la práctica del profesor para los datos de entrada específicos, en una subcarpeta llamada P*X*PROF donde *X* es el número de práctica.

  En *directorio_de_trabajo*/CompConProf/DatosEspecificos también se guarda un fichero para el alumno que se seleccionó en la ejecución. Este fichero se llama por tanto P*X*D*Y*.txt (si es un ejecutable que esta en el directorio de trabajo), o bien (exeGen)P*X*D*Y*.txt (si se seleccionó un ejecutable generado por la compilación del *.pas* asociado), donde *X* es el número de la práctica e *Y* es el *DNI* del alumno seleccionado.

  Estos ficheros almacenan la información sucesivamente de todas las comparaciones posibles realizadas para cada alumno concreto.

En cualquier caso *tipoGen* es Automatica (***GA***) o Manual (***GM***) dependiendo de la generación elegida. En este archivo se anota para cada alumno, el resultado de intentar realizar la comparación entre los ficheros de salida generados por los alumnos y los ficheros de salida generados por el ejecutable del profesor.

Un ejemplo de listado '*tipoGen*_Ejecutables.txt' sería:

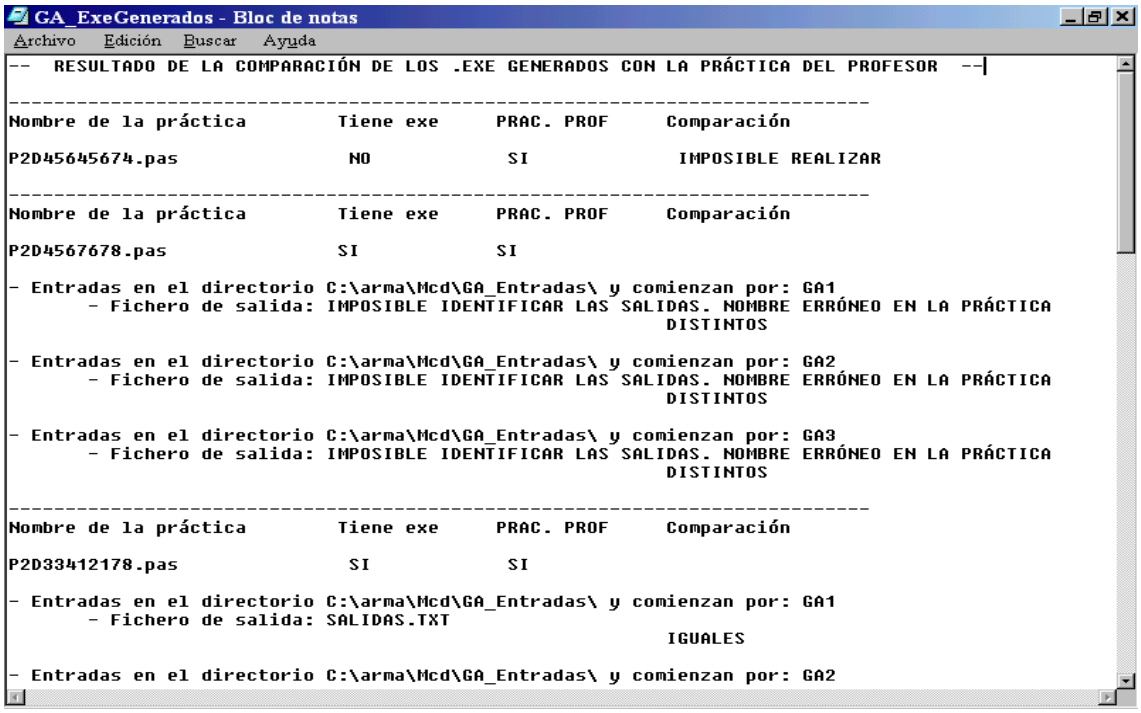


GA_Ejecutables - Bloc de notas

Archivo Edición Buscar Ayuda

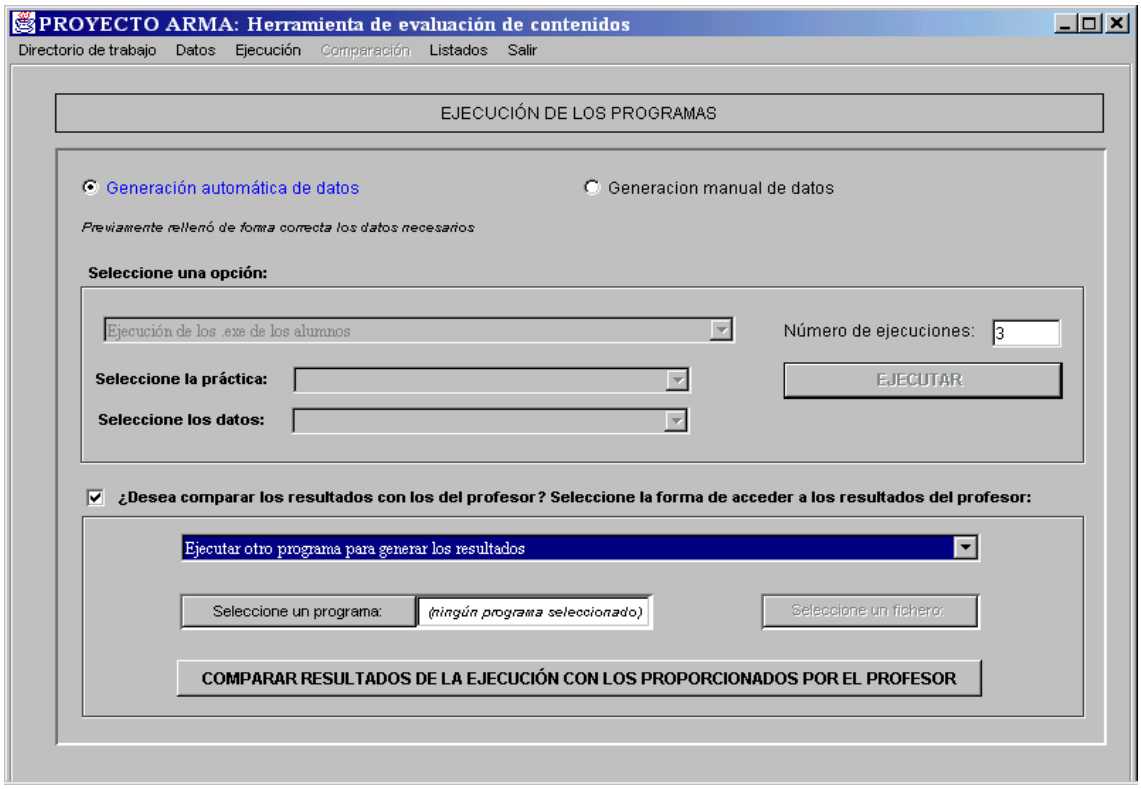
```
--  RESULTADO DE LA COMPARACIÓN DE LOS .EXE CON LA PRÁCTICA DEL PROFESOR  --

-------------------------------------------------------------------------------
Nombre de la práctica        Tiene exe     PRAC. PROF     Comparación

P2D45645674.pas               NO            SI             IMPOSIBLE REALIZAR

-------------------------------------------------------------------------------
Nombre de la práctica        Tiene exe     PRAC. PROF     Comparación

P2D4567678.pas                SI            SI

- Entradas en el directorio C:\arma\Mcd\GA_Entradas\ y comienzan por: GA1
       - Fichero de salida: IMPOSIBLE IDENTIFICAR LAS SALIDAS. NOMBRE ERRÓNEO EN LA PRÁCTICA
                                                              DISTINTOS

- Entradas en el directorio C:\arma\Mcd\GA_Entradas\ y comienzan por: GA2
       - Fichero de salida: IMPOSIBLE IDENTIFICAR LAS SALIDAS. NOMBRE ERRÓNEO EN LA PRÁCTICA
                                                              DISTINTOS

- Entradas en el directorio C:\arma\Mcd\GA_Entradas\ y comienzan por: GA3
       - Fichero de salida: IMPOSIBLE IDENTIFICAR LAS SALIDAS. NOMBRE ERRÓNEO EN LA PRÁCTICA
                                                              DISTINTOS

-------------------------------------------------------------------------------
Nombre de la práctica        Tiene exe     PRAC. PROF     Comparación

P2D33412178.pas               NO            SI             IMPOSIBLE REALIZAR

-------------------------------------------------------------------------------
Nombre de la práctica        Tiene exe     PRAC. PROF     Comparación

P2D76789098.pas               SI            SI

- Entradas en el directorio C:\arma\Mcd\GA_Entradas\ y comienzan por: GA1
       - Fichero de salida: SALIDAS.TXT
                                                              IGUALES

- Entradas en el directorio C:\arma\Mcd\GA_Entradas\ y comienzan por: GA2
       - Fichero de salida: SALIDAS.TXT
                                                              IGUALES

- Entradas en el directorio C:\arma\Mcd\GA_Entradas\ y comienzan por: GA3
       - Fichero de salida: SALIDAS.TXT
                                                              IGUALES
```

Un ejemplo de listado '*tipoGen*_ExeGenerados.txt' sería:

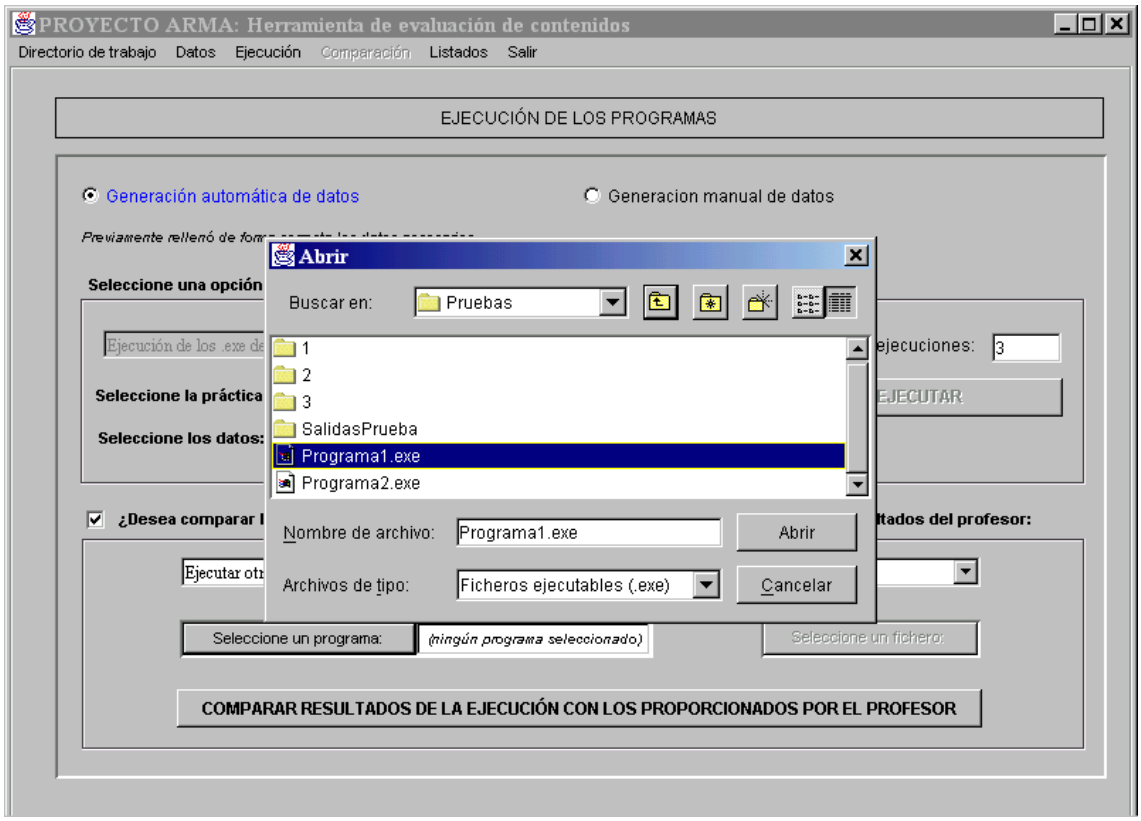



### *4.2.2.2.b.- Ejecutar un programa para obtener los resultados*

Esta opción permite seleccionar al profesor un programa distinto de la práctica del profesor para comparar con él las prácticas de los alumnos. La aplicación ejecutará dicho programa para las entradas seleccionadas en la última ejecución (que es sobre la que se quiere hacer la comparación) y comparará las salidas generadas, con las salidas de los alumnos.



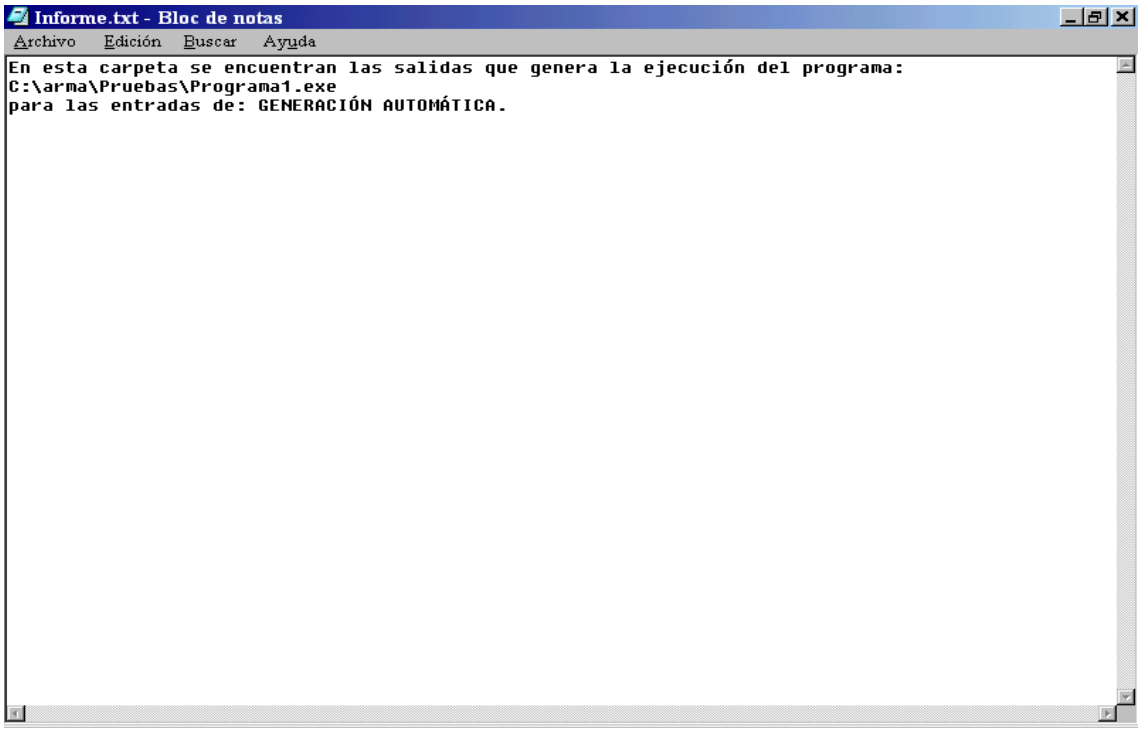
Como puede observarse cuando se selecciona esta opción se activa el botón "*Seleccione un programa*", el profesor debe pulsar sobre dicho botón y elegir el programa deseado.




Una vez seleccionado el programa el usuario debe pulsar el botón *COMPARAR RESULTADOS DE LA EJECUCIÓN CON LOS PROPORCIONADOS POR EL PROFESOR.*

La aplicación va guardando los resultados que se generan al ejecutar dichos programas. En la carpeta 'CompConProf" (que se encuentra en el directorio de trabajo) se generan diferentes subcarpetas con nombre *tipoGen*_SalidasProg*X*, siendo *tipoGen* Automatica (***GA***) o Manual (***GM***) dependiendo de la generación elegida, y *X* el número de programa, es decir como esta opción de comparar los resultados ejecutando un programa se puede realizar todas las veces que se desee en cualquier momento, la aplicación enumera los programas ejecutados para cada uno de los datos; dentro de cada una de estas subcarpetas se encuentran las salidas que generó la ejecución de dicho programa y un archivo de informe que indica el nombre del programa que se ejecutó.

Por ejemplo para el caso anterior en la ruta: *directorio_de_trabajo*/CompConProf/GA_SalidasProg1/ se encuentran las salidas generadas para el programa seleccionado: *Programa1.exe*, y el siguiente archivos de informe:




De nuevo la información que se guarda como resultado de la comparación, dependerá de la opción que se haya seleccionado en la última ejecución realizada. De esta forma se crean diferentes ficheros de texto en la carpeta 'CompConProf" (que se encuentra en el directorio de trabajo actual):

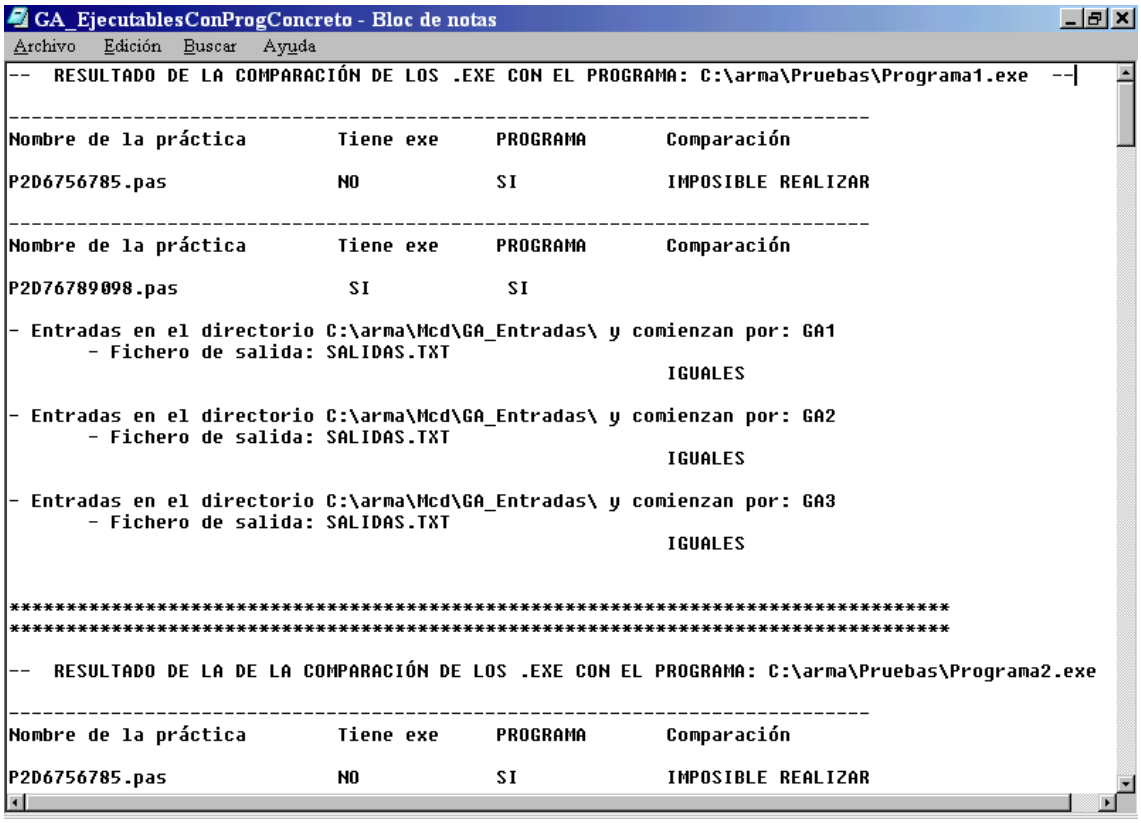
- *tipoGen*_EjecutablesConProgConcreto.txt (si la última ejecución realizada fue *'ejecución de los .exe de los alumnos'*)

- *tipoGen*_ExeGeneradosConProgConcreto.txt (si la última ejecución realizada fue *'compilación y ejecución de los programas'*)

- ambos ficheros de texto (*tipoGen*_EejecutablesConProgConcreto.txt y *tipoGen*_ExeGeneradosConProgConcreto.txt), si la última ejecución realizada fue *'ejecución del .exe y comparación con la compilación y la posterior ejecución de dicha práctica'*.

- *S*i la última ejecución realizada fue *'ejecución de la práctica de un único alumno con unos datos de entrada específicos'*, no se genera un listado genérico, sino un fichero de texto para la práctica concreta ejecutada con los datos de entrada específicos. En este caso en la carpeta 'CompConProf", se genera una subcarpeta llamada *DatosEspecificos* (si no estaba creada previamente) que guarda un fichero para el alumno que se seleccionó en la ejecución. Este fichero se llama por tanto P*X*D*Y*.txt (si es un ejecutable que esta en el directorio de trabajo), o bien (exeGen)P*X*D*Y*.txt (si se seleccionó un ejecutable generado por la compilación del *.pas* asociado), donde *X* es el número de la práctica e *Y* es el *DNI* del alumno seleccionado. Estos ficheros almacenan la información sucesivamente de todas las comparaciones posibles realizadas para cada alumno concreto.

Un ejemplo de listado '*tipoGen*_EjecutablesConProgConcreto.txt' sería:

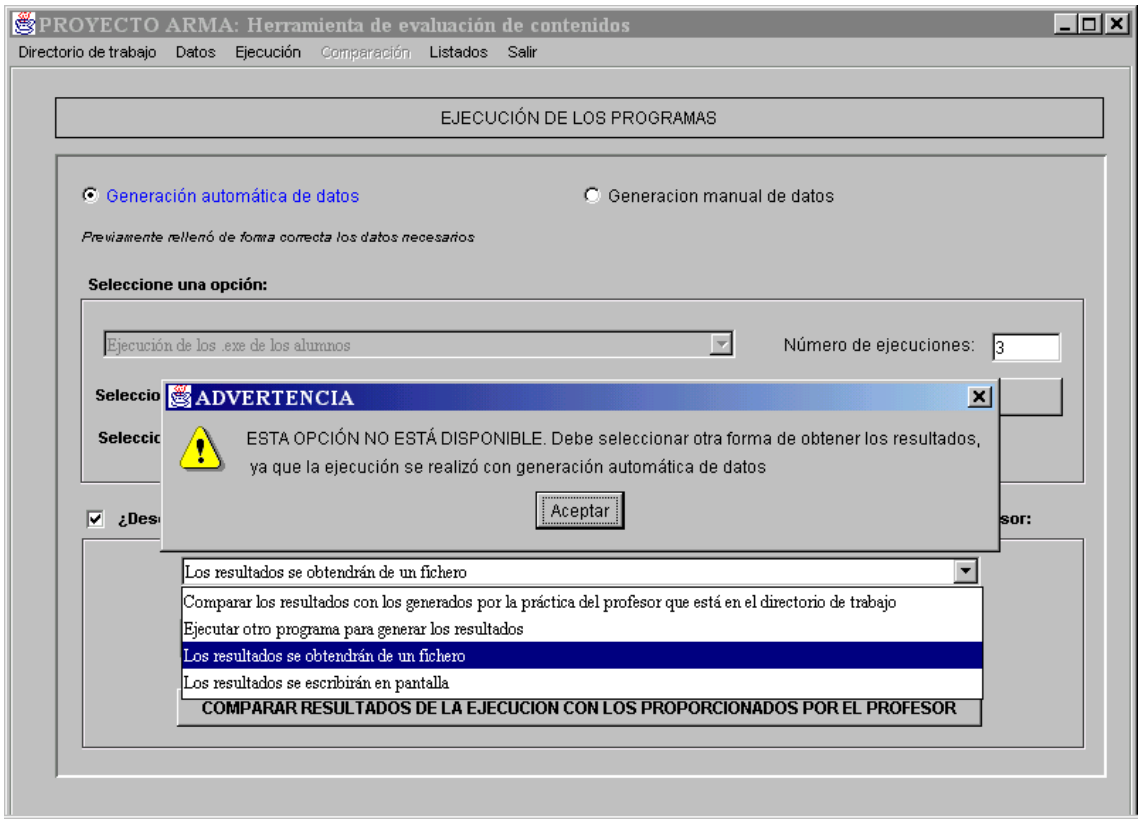



Para el caso '*tipoGen*_ExeGeneradosConProgConcreto.txt', el archivo sería análogo al anterior pero con la información relativa al intento de realizar la comparación con los ejecutables generados a partir de la compilación, en lugar de con los ejecutables dejados por los alumnos en el directorio de trabajo.

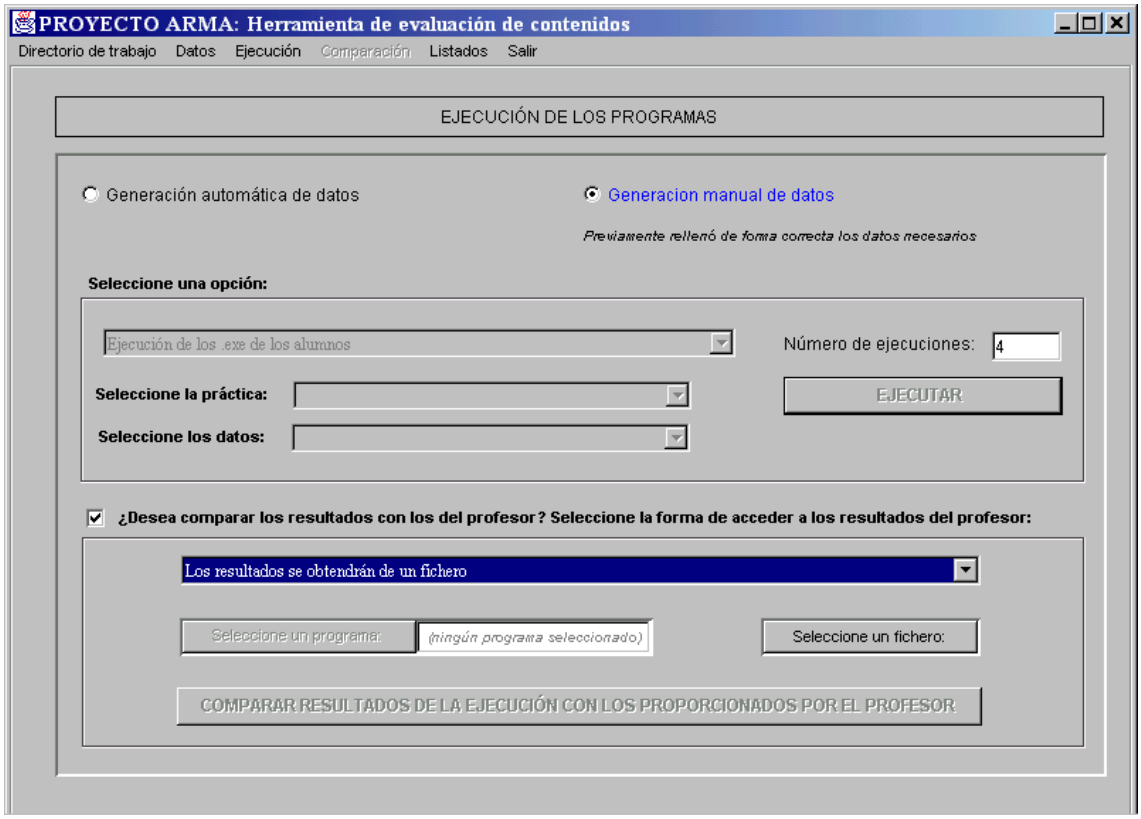
#### ***4.2.2.2.c.- Los resultados se obtendrán de un fichero***

Esta opción permite seleccionar al profesor unos ficheros concretos de salida para comparar con ellos las salidas de las prácticas de los alumnos.

Esta opción solo está permitida en el caso de que se haya realizado la ejecución con datos de entrada de *generación manual*. Si se intenta seleccionar esta opción para los datos de entrada de *generación automática* aparecerá el siguiente mensaje:

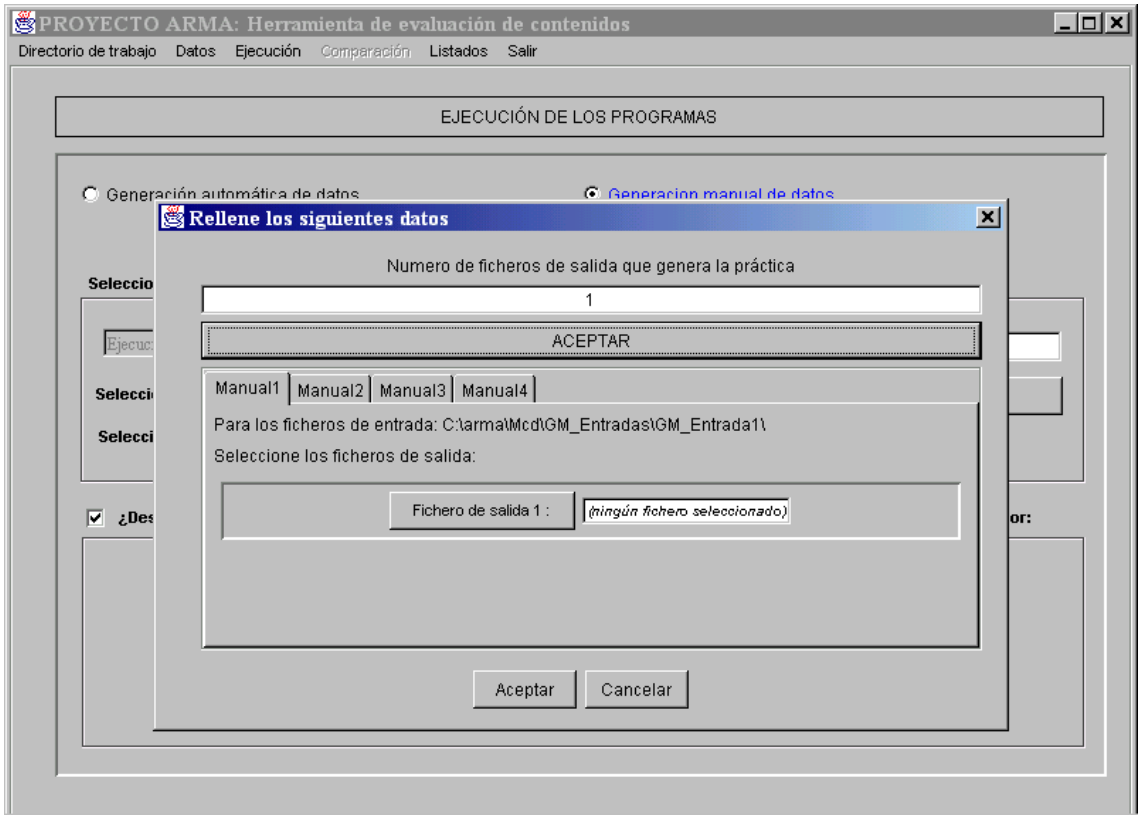



Seleccionamos por tanto la generación manual de datos, y realizamos las ejecuciones sobre dichos datos de entrada. De esta forma se nos permite elegir la comparación con el profesor obteniendo los resultados de un fichero:



Se puede observar que se ha activado el botón "*Seleccione un fichero*"; el usuario debe pinchar sobre dicho botón; así le aparecerá una pantalla donde debe rellenar, para cada una de las generaciones que se ejecutaron, los ficheros de salida con los que quiere que se realicen la comparaciones. El usuario debe tener en cuenta que los ficheros de salida seleccionados deben tener el mismo nombre que los que generan las prácticas de los alumnos, ya que en otro caso la aplicación intentará realizar la comparación pero es posible que no pueda identificar las salidas del profesor con las del alumno (dicha información se guardará en los nuevos ficheros de texto generados). En cualquier caso la aplicación informa al usuario de cualquier error que se pueda producir cuando el usuario rellena los datos, por ejemplo que deje algún campo sin rellenar.

La pantalla para rellenar los datos que aparece al pulsar sobre el botón "*Seleccione un fichero*" es de la siguiente forma:




Cuando el usuario haya terminado de seleccionar todos los ficheros de salida, debe pulsar el botón *Aceptar* de la pantalla rellenar los datos, y posteriormente el botón *COMPARAR LOS RESULTADOS DE LA EJECUCIÓN CON LOS PROPORCIONADOS POR EL PROFESOR* de la pantalla principal.

De nuevo la información que se guarda como resultado de la comparación, dependerá de la opción que se haya seleccionado en la última ejecución realizada. De esta forma se crean diferentes ficheros de texto en la carpeta 'CompConProf" (que se encuentra en el directorio de trabajo actual):

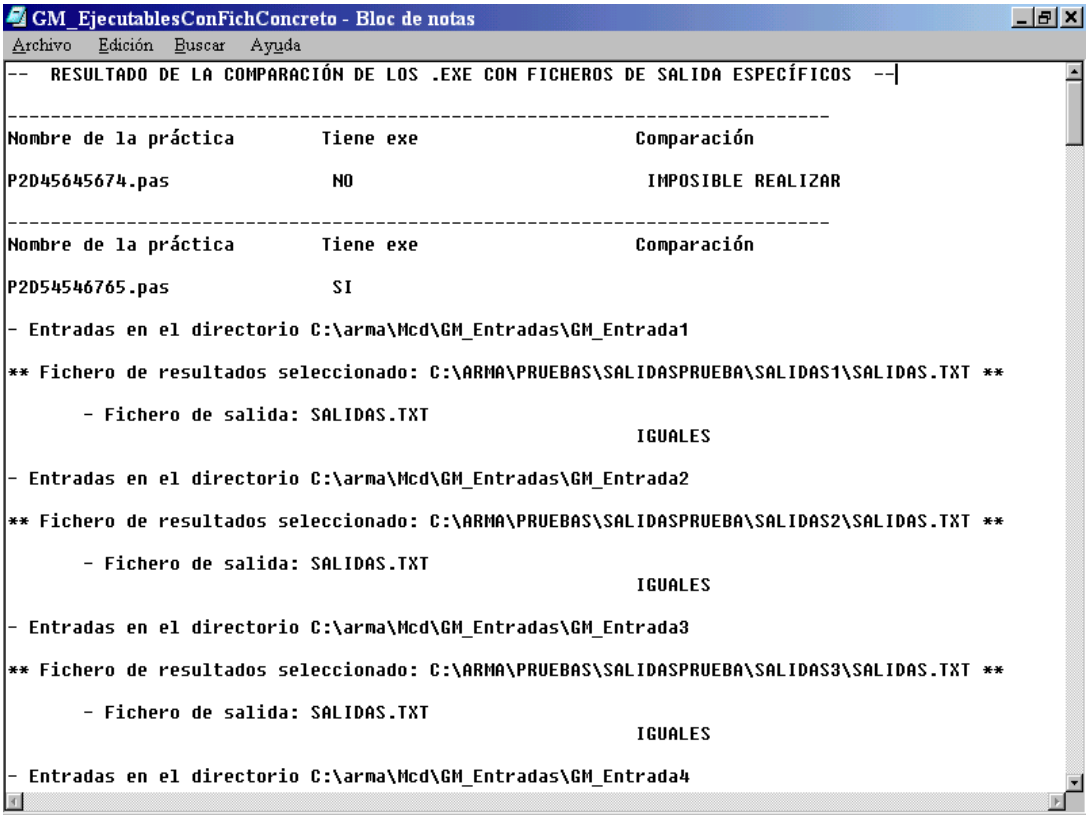
- GM_EjecutablesConFichConcreto.txt (si la última ejecución realizada fue *'ejecución de los .exe de los alumnos'*)

- GM_ExeGeneradosConFichConcreto.txt (si la última ejecución realizada fue *'compilación y ejecución de los programas'*)

- ambos ficheros de texto (GM_EejecutablesConFichConcreto.txt y GM_ExeGeneradosConFichConcreto.txt), si la última ejecución realizada fue *'ejecución del .exe y comparación con la compilación y la posterior ejecución de dicha práctica'*.

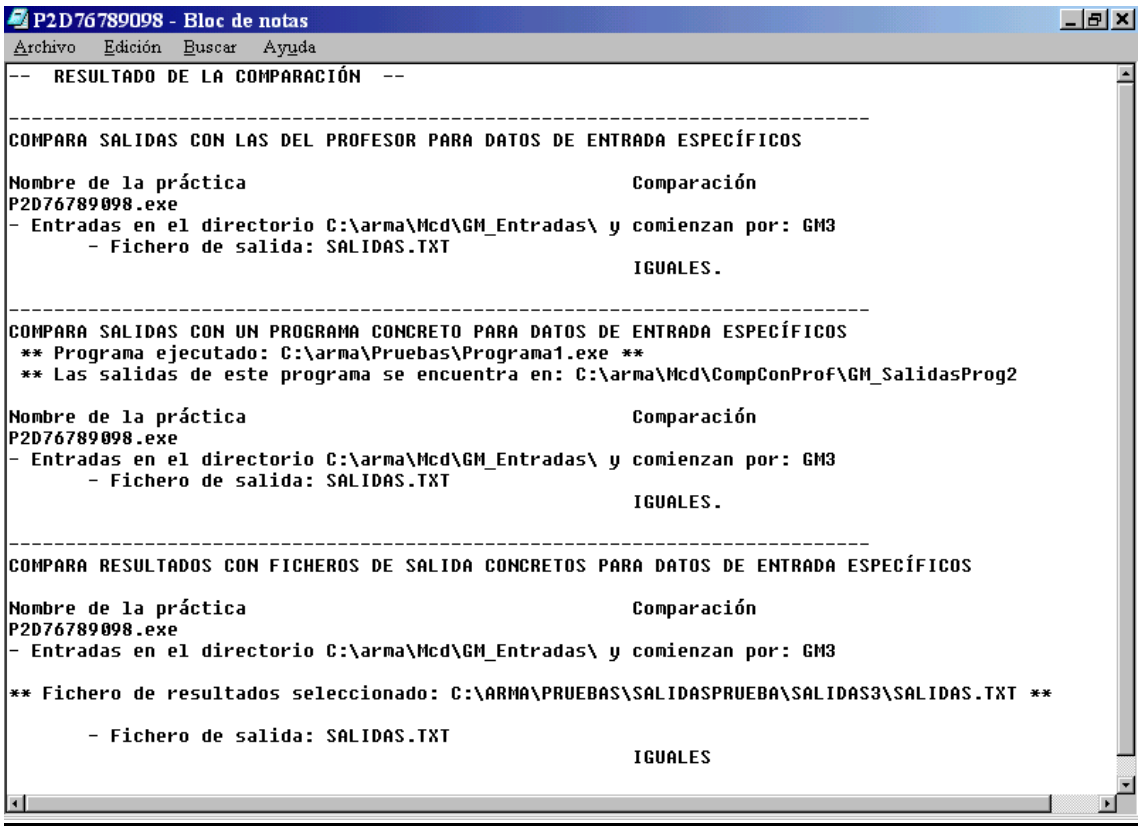
- *S*i la última ejecución realizada fue *'ejecución de la práctica de un único alumno con unos datos de entrada específicos'*, no se genera un listado genérico, sino un fichero de texto para la práctica concreta ejecutada con los datos de entrada específicos. En este caso en la carpeta 'CompConProf', se genera una subcarpeta llamada *DatosEspecificos* (si no estaba creada previamente) que guarda un fichero para el alumno que se seleccionó en la ejecución. Este fichero se llama por tanto P*X*D*Y*.txt (si es un ejecutable que esta en el directorio de trabajo), o bien (exeGen)P*X*D*Y*.txt (si se seleccionó un ejecutable generado por la compilación del *.pas* asociado), donde *X* es el número de la práctica e *Y* es el *DNI* del alumno seleccionado. Estos ficheros almacenan la información sucesivamente de todas las comparaciones posibles realizadas para cada alumno concreto.

Un ejemplo de listado 'GM_EjecutablesConFichConcreto.txt' sería:



Para el caso 'GM_ExeGeneradosConFichConcreto.txt', el archivo sería análogo al anterior pero con la información relativa al intento de realizar la comparación con los ejecutables generados a partir de la compilación, en lugar de con los ejecutables dejados por los alumnos en el directorio de trabajo.

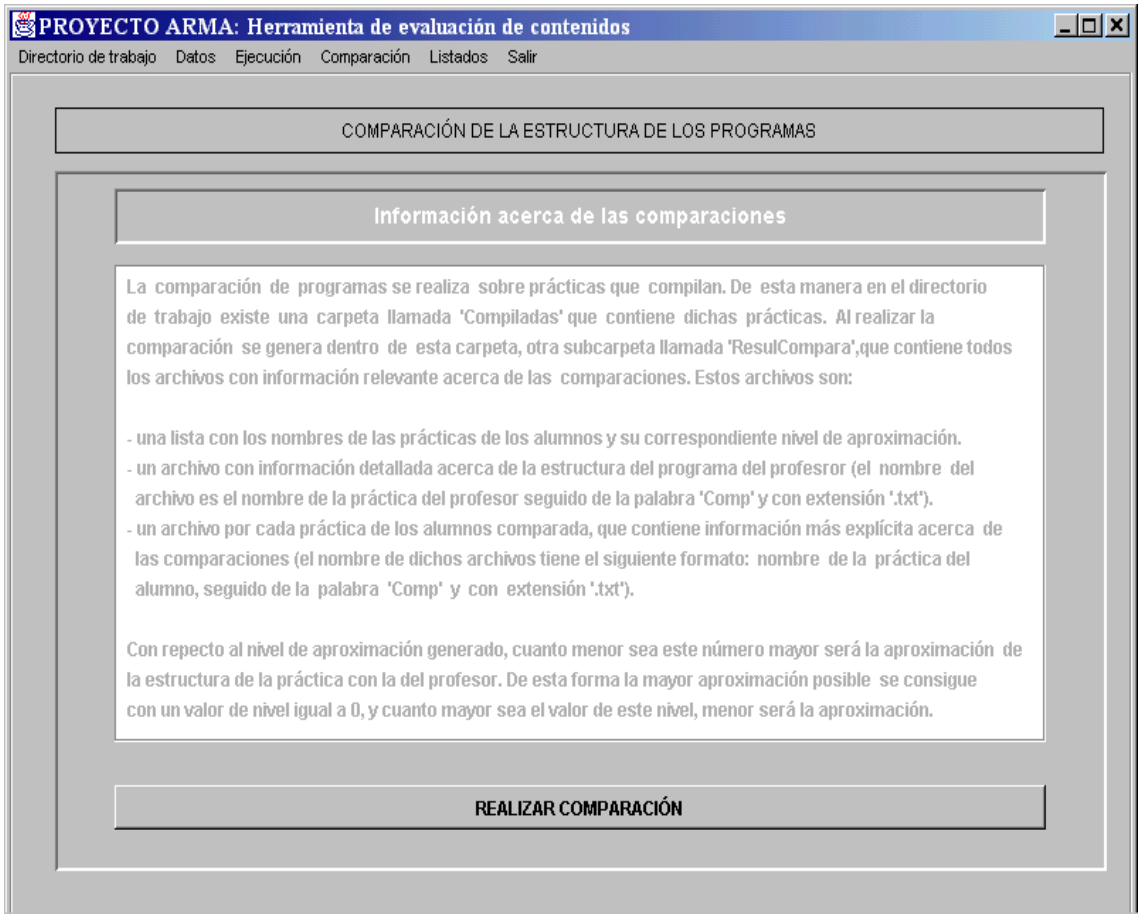
Un ejemplo de fichero generado por la comparación cuando se selecciona la ejecución de la práctica de un único alumno con unos datos de entrada específicos, en este caso el alumno seleccionado fue P276789098 por eso el nombre del fichero es 'P2D76789098.txt', ubicado como se ha dicho anteriormente en *directorio_de_trabajo*/CompConProf/DatosEspecificos, sería el siguiente:




Puede observarse como en este fichero se va almacenando la información sucesivamente, así pues para este alumno concreto y con los datos de entrada *GM_Entrada3*, primero se realizó una comparación con la práctica del profesor, después se realizó una comparación ejecutando un programa específico para obtener los resultados, y por último se realizó una comparación seleccionado un fichero de salida concreto.

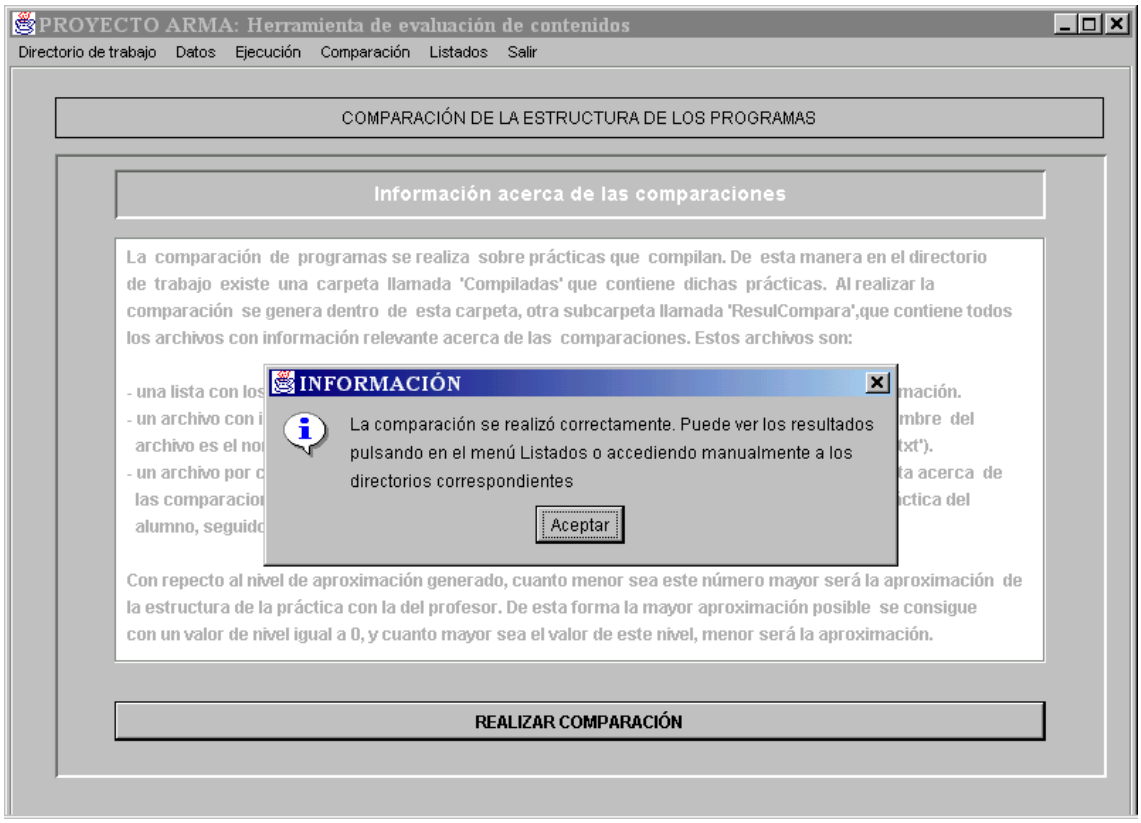
## 4.2.3.- Comparación de la estructura de los programas

El menú para acceder a esta pantalla solo se activa una vez que se han compilado los programas, puesto que solo tiene sentido comparar las estructuras de los programas de los alumnos con la estructura del programa del profesor, para aquellas prácticas que al menos compilan. La aplicación por tanto controla esto, activando únicamente el menú de comparación, cuando se han compilado (como mínimo una vez). La pantalla de comparación presenta las siguientes características:




El usuario solo debe preocuparse de pulsar el botón "*REALIZAR COMPARACIÓN*".
Si la comparación no se realiza con éxito, el sistema informa al usuario con un mensaje de error.

Por el contrario si la comparación se realiza con éxito la aplicación muestra lo siguiente:




En la carpeta ‘Compiladas’ (que se encuentra en el directorio de trabajo) se genera una nueva subcarpeta llamada ‘ResulCompara’ que guarda la información (en ficheros de texto) acerca de la comparación de las estructuras de los programas:

- Por un lado, contiene un listado llamado ‘ListaComp.txt’, con los nombres de todos los alumnos (cuyas prácticas compilan), y el nivel de aproximación de cada una de las prácticas con la práctica del profesor. Cuánto menor sea este nivel, mayor será la aproximación de la estructura de la práctica del alumno con la estructura de la práctica del profesor.

El nivel de aproximación se mide de la siguiente manera:

- La *diferencia máxima de número de variables* que puede tener el alumno declaradas en comparación con las que tiene el profesor es *2*.

- La *diferencia máxima de número de operadores* utilizados que puede tener el alumno en comparación con los que tiene el profesor es *2*.

- La *diferencia máxima de número de líneas de código* que puede tener el alumno en comparación con las que tiene el profesor es *2*.

- *No se permite diferencia entre el número de instrucciones de control* que tiene el alumno y el numero que tiene el profesor.

- *No se permite diferencia entre el número de subprogramas definidos* que tiene el alumno y el numero que tiene el profesor.

- Deben coincidir las inicializaciones de las variables dentro del programa.

- Deben coincidir las instrucciones de control teniendo en cuenta que:

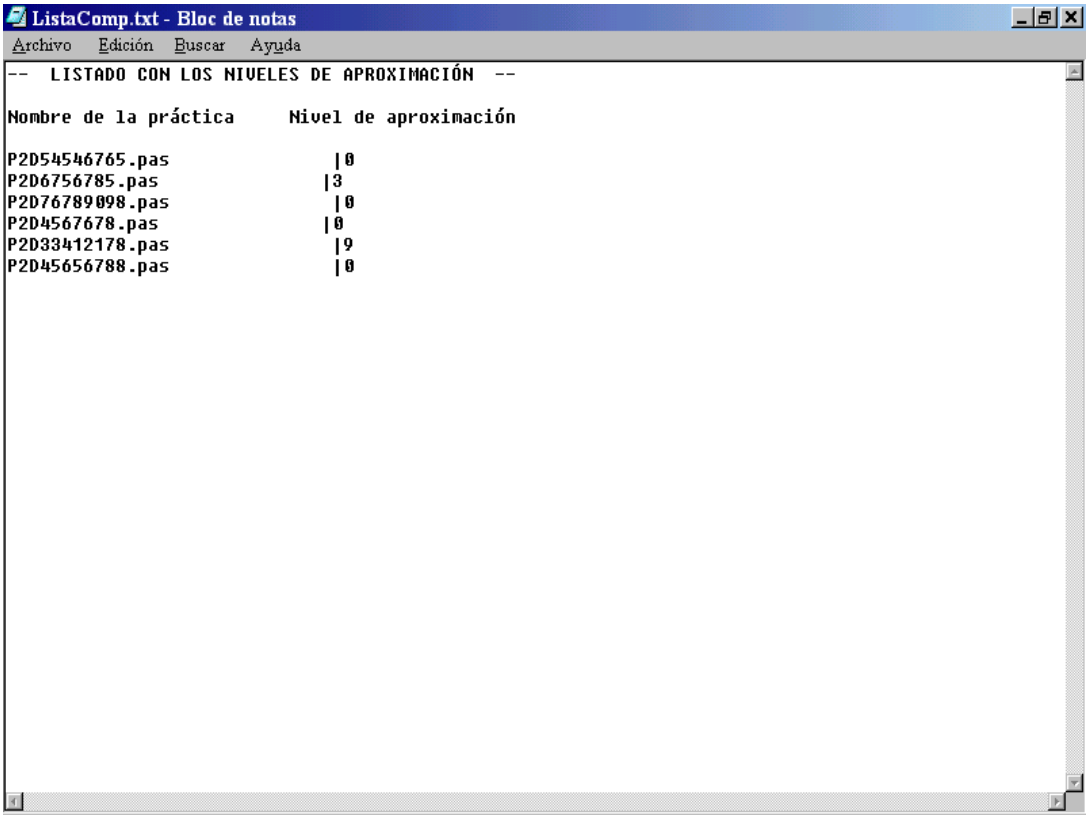
  - se ha supuesto que la instrucción 'IF' no tiene instrucción equivalente.

  - se ha supuesto que las instrucciones 'WHILE', 'FOR' y 'REPEAT' son equivalentes.

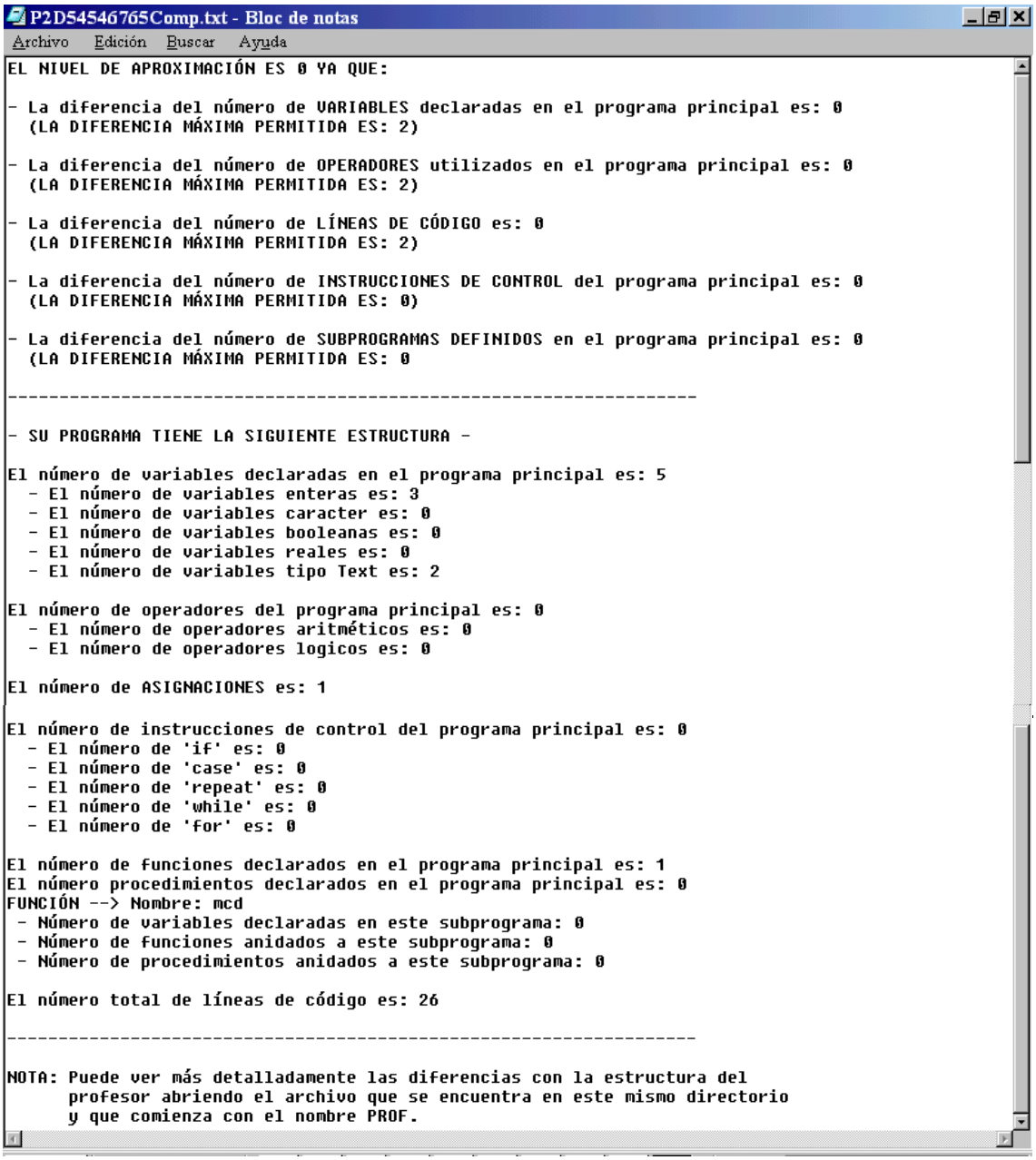
  - se ha supuesto que la instrucción 'CASE' no tiene instrucción equivalente.

Según esto, se suman las diferencias y cuando el número de estas supera el máximo permitido se penaliza con 5 puntos.

Un ejemplo del listado que se genera, sería el siguiente:



- Por otro lado, en *directorio_de_trabajo*/Compiladas/ResulCompara se guarda también un fichero de texto para cada alumno (llamado 'P*X*D*Y*Comp.txt' siendo *X* el número de práctica e *Y* el *DNI* del alumno), que contiene información más detallada acerca de la comparación, y de por qué ese nivel de aproximación que aparece en los listados. Para el caso anterior, seleccionamos por ejemplo el fichero correspondiente al primer alumno que aparece en la lista, y su contenido sería el siguiente:



P2D54546765Comp.txt - Bloc de notas

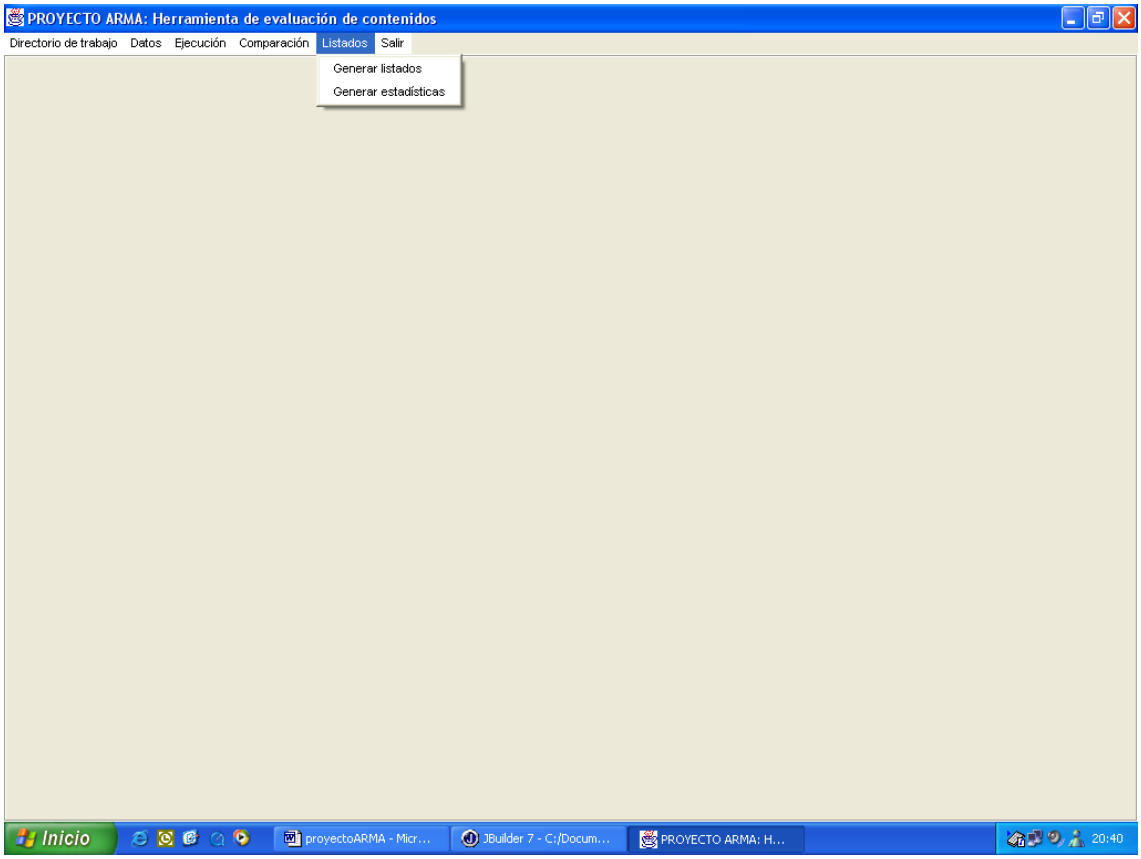
Archivo Edición Buscar Ayuda

```
EL NIVEL DE APROXIMACIÓN ES 0 YA QUE:

- La diferencia del número de VARIABLES declaradas en el programa principal es: 0
  (LA DIFERENCIA MÁXIMA PERMITIDA ES: 2)

- La diferencia del número de OPERADORES utilizados en el programa principal es: 0
  (LA DIFERENCIA MÁXIMA PERMITIDA ES: 2)

- La diferencia del número de LÍNEAS DE CÓDIGO es: 0
  (LA DIFERENCIA MÁXIMA PERMITIDA ES: 2)

- La diferencia del número de INSTRUCCIONES DE CONTROL del programa principal es: 0
  (LA DIFERENCIA MÁXIMA PERMITIDA ES: 0)

- La diferencia del número de SUBPROGRAMAS DEFINIDOS en el programa principal es: 0
  (LA DIFERENCIA MÁXIMA PERMITIDA ES: 0

-----------------------------------------------------------------

- SU PROGRAMA TIENE LA SIGUIENTE ESTRUCTURA -

El número de variables declaradas en el programa principal es: 5
  - El número de variables enteras es: 3
  - El número de variables caracter es: 0
  - El número de variables booleanas es: 0
  - El número de variables reales es: 0
  - El número de variables tipo Text es: 2

El número de operadores del programa principal es: 0
  - El número de operadores aritméticos es: 0
  - El número de operadores logicos es: 0

El número de ASIGNACIONES es: 1

El número de instrucciones de control del programa principal es: 0
  - El número de 'if' es: 0
  - El número de 'case' es: 0
  - El número de 'repeat' es: 0
  - El número de 'while' es: 0
  - El número de 'for' es: 0

El número de funciones declarados en el programa principal es: 1
El número procedimientos declarados en el programa principal es: 0
FUNCIÓN --> Nombre: mcd
 - Número de variables declaradas en este subprograma: 0
 - Número de funciones anidados a este subprograma: 0
 - Número de procedimientos anidados a este subprograma: 0

El número total de líneas de código es: 26

-----------------------------------------------------------------

NOTA: Puede ver más detalladamente las diferencias con la estructura del
      profesor abriendo el archivo que se encuentra en este mismo directorio
      y que comienza con el nombre PROF.
```

## 4.2.4.- Listados

Con los datos obtenidos se pueden ver y generar listados y estadísticas. En el menú principal se puede pulsar sobre *Listados*, donde se presentan las opciones disponibles. Éstas son

4.2.4.1.- Generar Listados
4.2.4.2.- Generar Estadísticas




### 4.2.4.1.- Generación de listados

Si seleccionamos la opción *Generar Listados*, se nos presenta la siguiente pantalla, donde se encuentran los listados disponibles. A la izquierda de la pantalla hay una tabla para mostrar los alumnos por grupos; y a la derecha de la pantalla se encuentran los tipos de listados disponibles:

- Prácticas correctas
- Prácticas que compilan
- Prácticas que se ejecutan
- Errores encontrados en las prácticas (esta opción no esta disponible en esta versión del proyecto)
- Nivel de aproximación al código del profesor

La opción *Prácticas correctas* mostrará la información sobre la corrección de las prácticas para los alumnos seleccionados. Hemos tenido en cuenta dos puntos diferentes en cuanto a la corrección de una práctica:

- Que la ejecución del *.exe* dejado por el alumno coincida con la ejecución del *.exe* generado a partir del *.pas* del alumno. Si el alumno no ha dejado el ejecutable, no se considerará correcta la práctica.

- Que la ejecución del *.exe* dejado por el alumno coincida con la ejecución del *.exe* del profesor. Si el alumno no ha dejado el ejecutable o si ha dejado un ejecutable pero no un *.pas*, la práctica no se considerará correcta.

La opción *Prácticas que compilan* mostrará la información sobre si las prácticas de los alumnos seleccionados compilan o no. Para poder ver esta información es necesario que se hayan compilado previamente las prácticas.

La opción *Prácticas que se ejecutan* mostrará la información sobre si las prácticas de los alumnos ejecutan o no, teniendo en cuenta que se pueden haber ejecutado con datos generados automáticamente o con datos generado manualmente. También se tiene en cuenta que la práctica se puede ejecutar varias veces para distintos datos de entrada.

La opción *Errores encontrados* no está disponible en esta versión, se trataría de mostrar los errores encontrados durante la compilación y ejecución de las prácticas de los alumnos.

La opción *Niveles de aproximación* muestra el nivel de aproximación al código del profesor de cada alumno seleccionado.

Debajo de estas opciones hay una pestaña para elegir sobre qué resultados se quieren generar los listados:

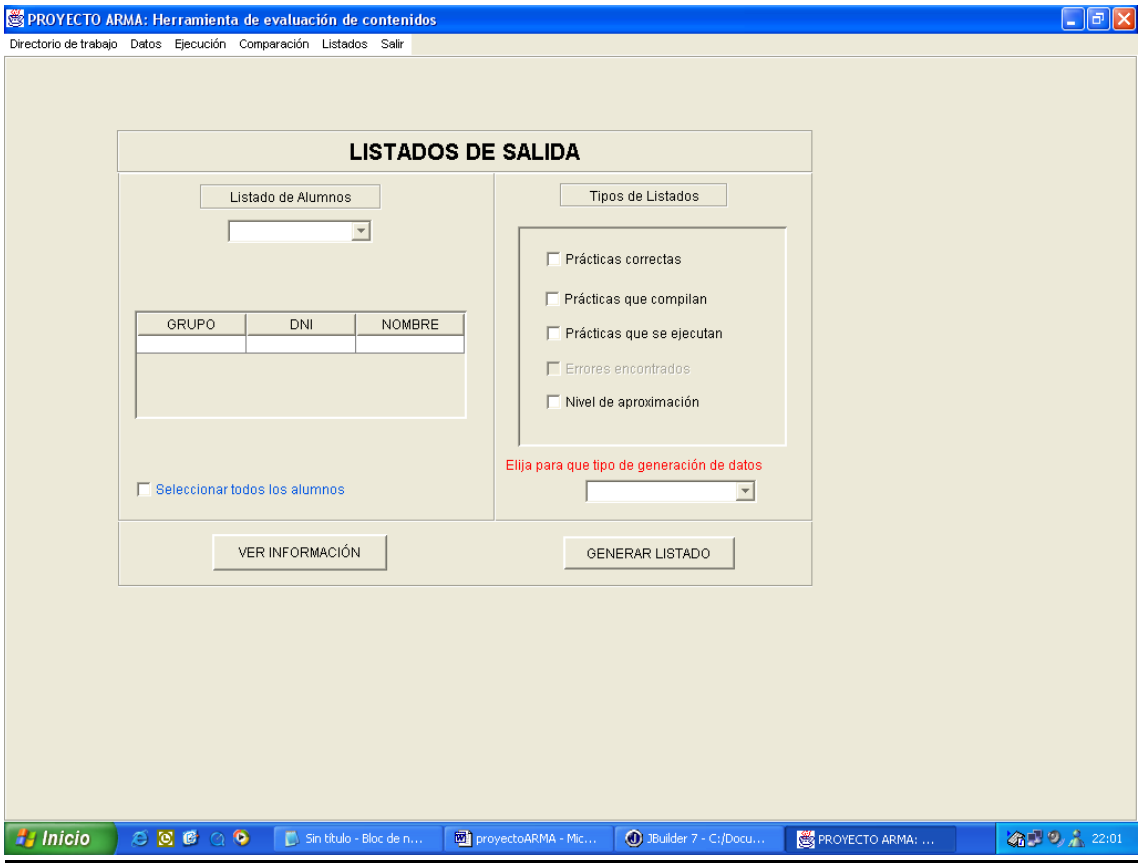
- Sobre las prácticas ejecutadas mediante generación aleatoria de datos

- Sobre las prácticas ejecutadas mediante generación manual de datos

En la parte inferior de la pantalla hay dos botones:

- El botón *Ver Información*: presenta por pantalla el listado.

- El botón *Generar Listado*: genera un listado con formato Excel.

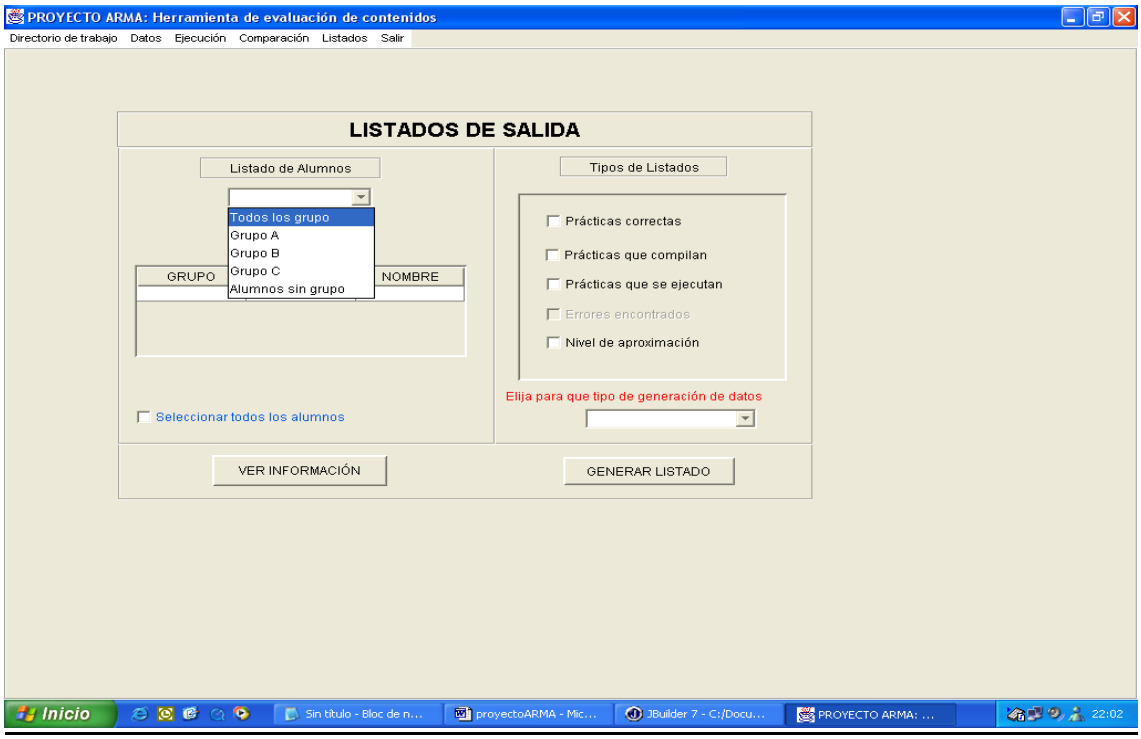



### *4.2.4.1.a. – Mostrar un listado por pantalla*

Para mostrar un listado por pantalla se procede de la siguiente manera. Primero hay que mostrar y elegir los alumnos sobre los que se quiere mostrar el listado. Para ello, en la pestaña Listado de Alumnos se debe elegir primero qué grupo de alumnos se quiere mostrar en la tabla:

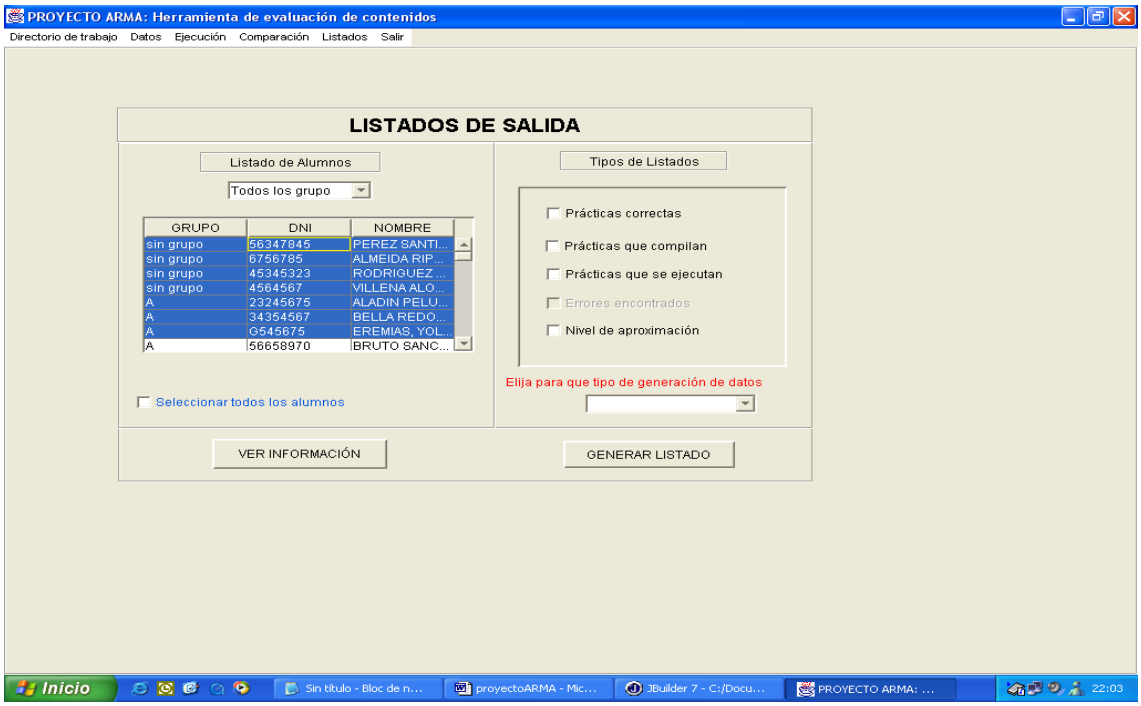
- Todos los grupos
- Grupo A
- Grupo B
- Grupo C
- Alumnos sin grupo

Una vez elegido el grupo, aparecerán en la tabla los nombres, DNIs y grupo de los alumnos.

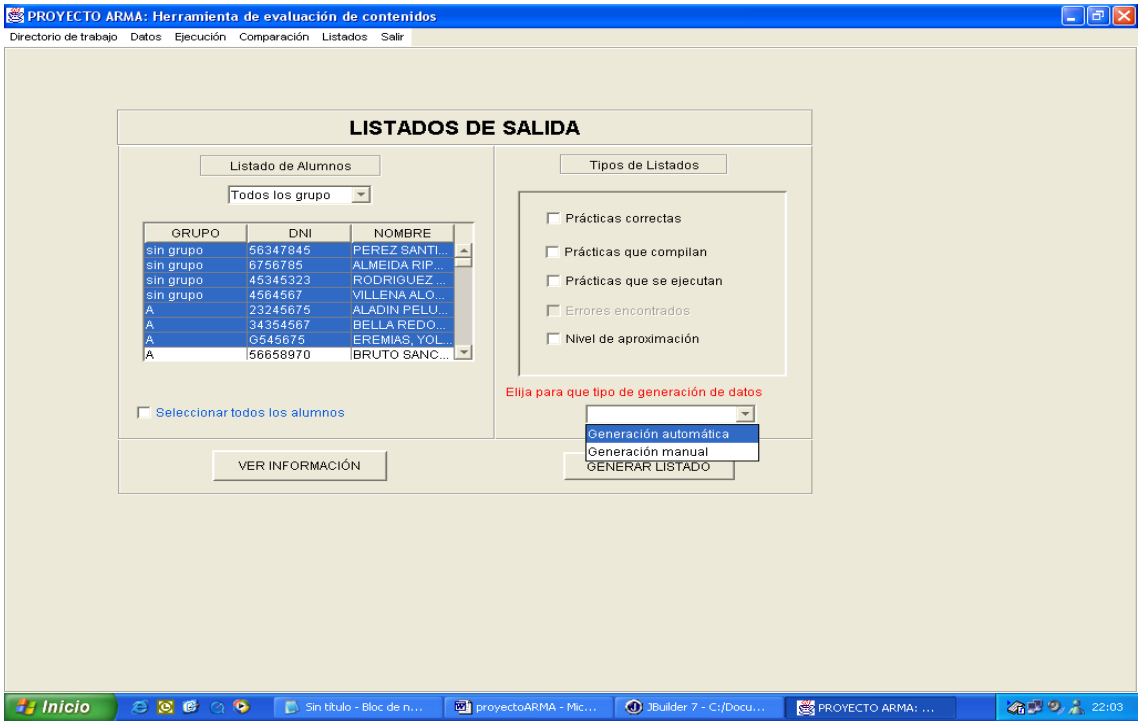



En este caso hemos elegido la opción *Todos los grupos*. En la tabla se muestran los alumnos de todos los grupos. Una vez que tenemos en la tabla a los alumnos, podemos seleccionar el grupo de alumnos o el alumno sobre el que se quiere mostrar el listado.

Seleccionando el checkbox *Seleccionar todos los alumnos*, se seleccionan todos los alumnos de la tabla. Pero podemos seleccionarlos también de forma manual.




Después de seleccionar a los alumnos debemos seleccionar sobre qué datos de las ejecuciones queremos ver la información: los datos de las prácticas ejecutadas con datos generados automáticamente o manualmente.



En este ejemplo hemos seleccionado la generación automática. Una vez indicados los alumnos y el tipo de generación de datos, podemos seleccionar el tipo de listado que queremos. Se puede elegir solo un tipo o varios, dependiendo de nuestras preferencias. En este ejemplo elegiremos todas las opciones posibles: si las prácticas compilan, si ejecutan, si son correctas y el nivel de aproximación al código del profesor. Se puede elegir cualquier combinación de opciones.


Una vez que tenemos todo (alumnos, tipo de generación de datos, tipos de listado) pulsaremos sobre el botón *VER INFORMACIÓN*.

Al pulsar sobre este botón, aparecerá otra ventana con los alumnos seleccionados y la información correspondiente. Aparece una nueva ventana con la información requerida para cada alumno seleccionado.

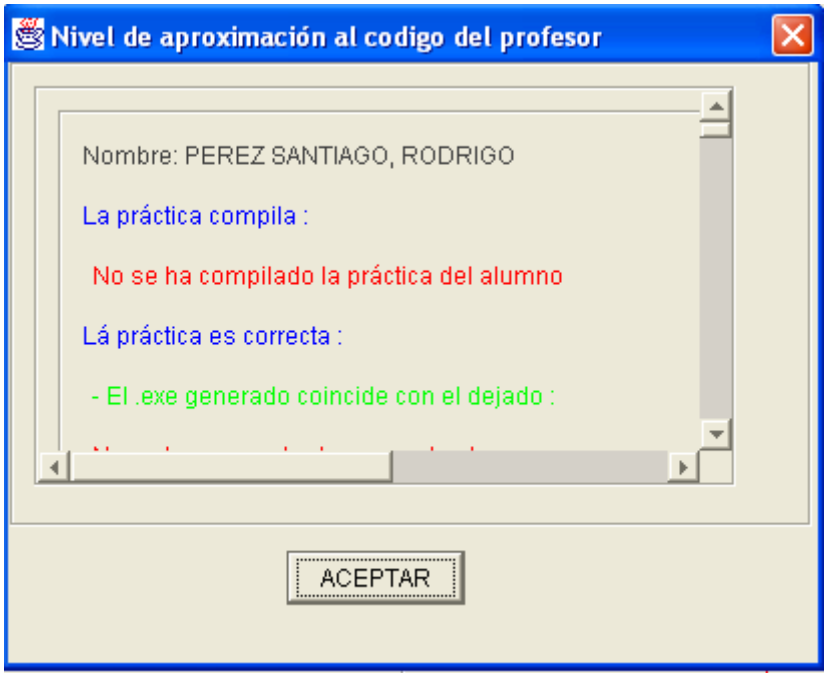
En nuestro caso, al haber seleccionado todas las opciones disponibles aparecerá la información disponible para el alumno respecto a todas estas opciones. Cada alumno sale en un recuadro distinto. El tipo de información que se muestra va en color azul, y el resultado para el alumno, en color rojo.

Para el caso de si la práctica compila, aparece una etiqueta azul *La práctica compila* a continuación se muestra en rojo el resultado. Si la práctica compila se mostrará el mensaje *Compila* en caso contrario *No compila*. En el caso de que la práctica no se haya compilado, o la práctica del alumno no haya sido entregada al profesor, también se mostrará un mensaje informando de dicha situación.

Para el caso de si la práctica ejecuta, aparece, o bien un mensaje en rojo informando de que la práctica no ha sido ejecutada o el alumno no entregó ejecutable, o bien un mensaje para pulsar botón que nos llevará a otra ventana con la información sobre la ejecución de la práctica de ese alumno.

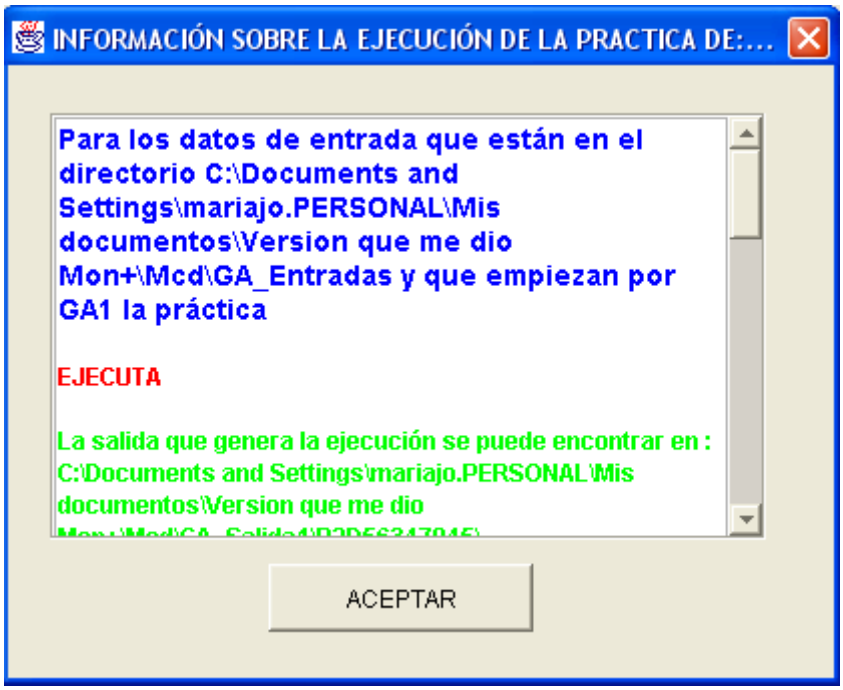
En el caso del nivel de aproximación, aparece, igualmente que en el caso de que si compila, una etiqueta en azul *Nivel*; a continuación en rojo aparecerá bien el nivel de aproximación al código del profesor, bien un mensaje informando de que la práctica del alumno no se ha comparado con la del profesor.

En el caso de si la práctica es correcta aparece la información de forma similar a los casos anteriores. Como hemos comentado anteriormente, la corrección de la práctica la hemos abordado desde dos puntos de vista: si la ejecución del ejecutable entregado por el alumno y la ejecución del ejecutable generado por el archivo fuente dejado coinciden y si la ejecución del ejecutable del alumno coincide con la del ejecutable del profesor. En la ventana, para cada alumno aparecerá la información desde cada uno de estos dos puntos de vista.



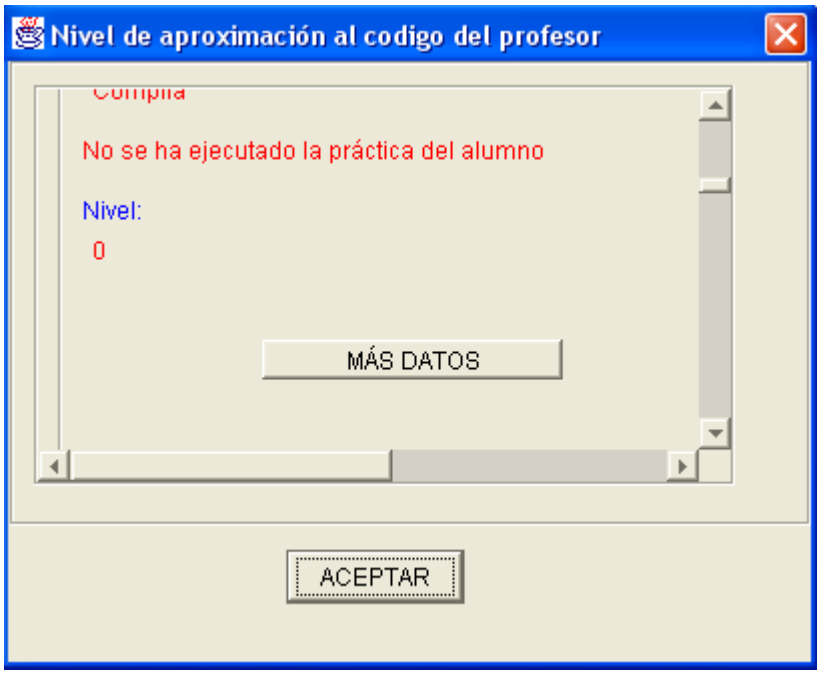

Moviendo las barras de desplazamiento se puede ver la información para cada alumno. Si se pulsa el botón *ACEPTAR* se cierra la ventana.

Como vemos en el dibujo, la práctica del alumno no se ha compilado; pero en cambio si se ha ejecutado, eso significa que el alumno no entregó el *.pas*. Aparece un botón *Datos Ejecución* que si pulsamos sobre él e abrirá otra ventana con la información sobre la ejecución del *.exe* del alumno con los diferentes datos generados según la forma elegida en el panel.

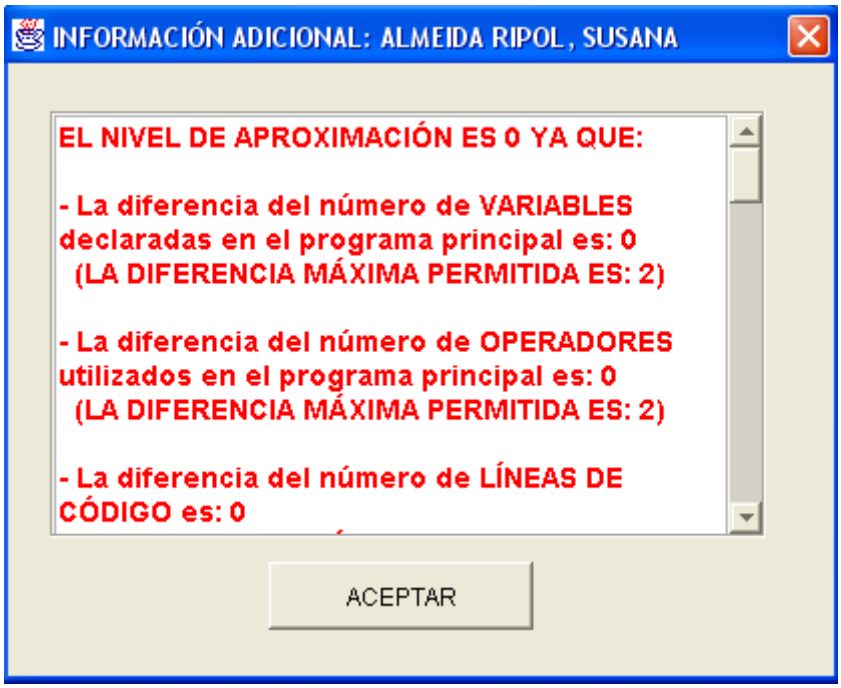



Moviendo las barras de desplazamiento podremos ver toda la información sobre la ejecución de la práctica del alumno.

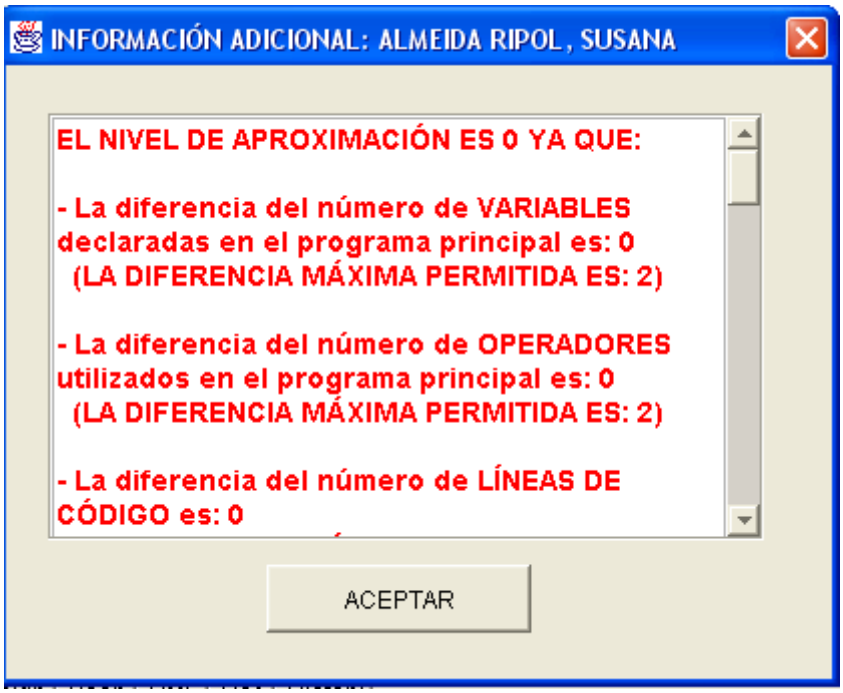
Podemos ver también información adicional sobre el nivel de aproximación al código del profesor, pulsando sobre el botón *MÁS DATOS* debajo de donde está la información sobre el nivel del alumno.




Para este alumno, que tiene nivel 0, pulsando en el botón *Más Datos* podremos obtener la justificación de dicho nivel.



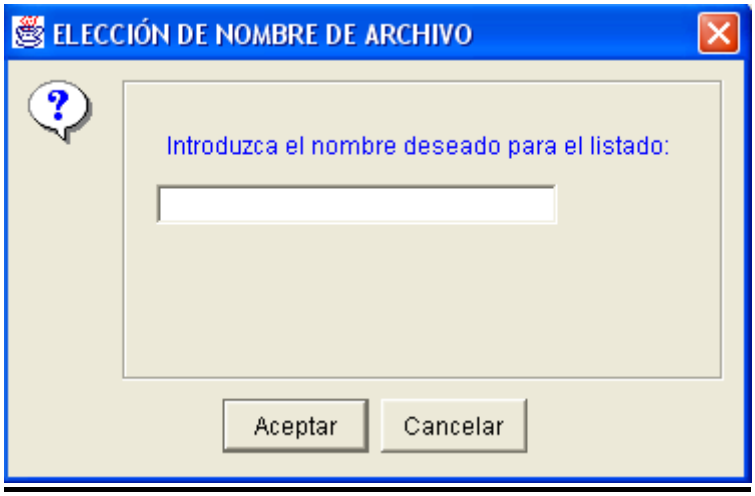

Moviendo las barras de desplazamiento podremos leer toda la información. Si pulsamos el botón *ACEPTAR* la ventana se cerrará.



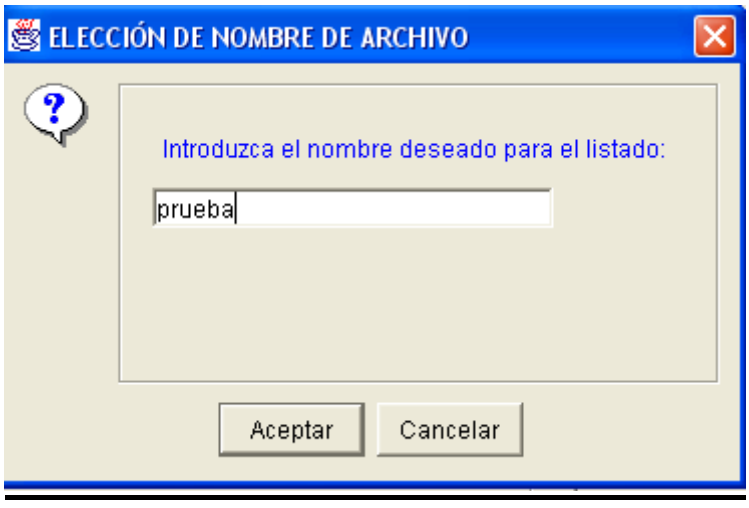
#### *4.2.4.1.b.- Generar un listado*

Para generar un listado, se procede como hemos descrito en el apartado anterior, pero esta vez, pulsaremos sobre el botón *Generar Listado*.

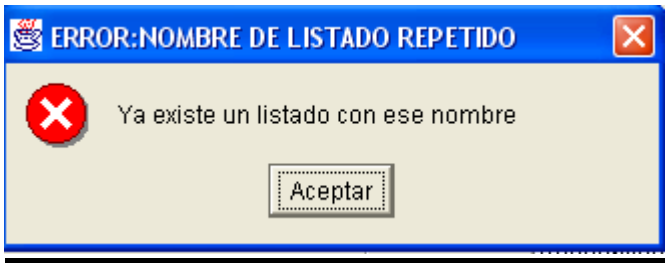
Aparecerá una ventana donde se deberá introducir el nombre que queremos para el listado:



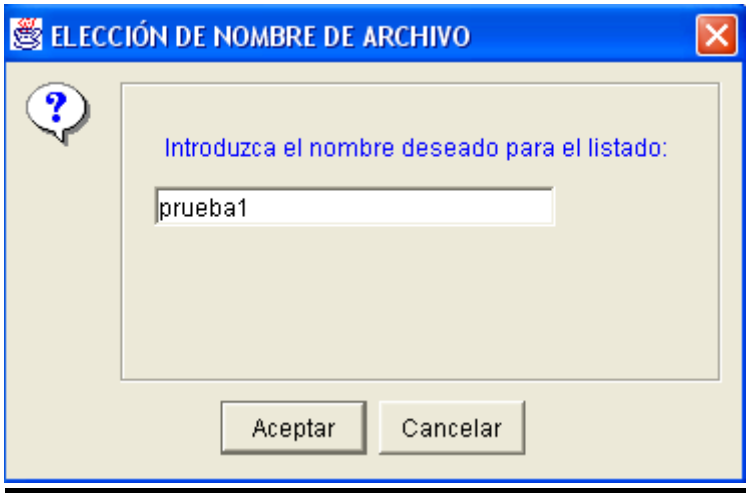

Introducimos el nombre que queremos. Los listados generados se guardarán en el directorio de trabajo en una carpeta llamada *Listados*. Si existiera ya un listado con el nombre elegido en esa carpeta, el sistema informará de ello y el listado no se creará. En ese caso se deberá introducir un nombre distinto que no exista ya.



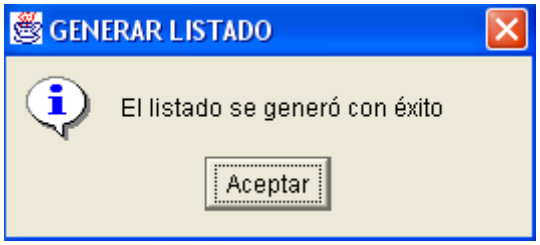

En este caso, si pulsamos aceptar comprobamos que ya existe otro listado con el nombre *prueba*. Debemos en este caso volver a pulsar el botón *Generar Listado*, con lo cual nos volverá a aparecer el diálogo para introducir el nombre del listado.



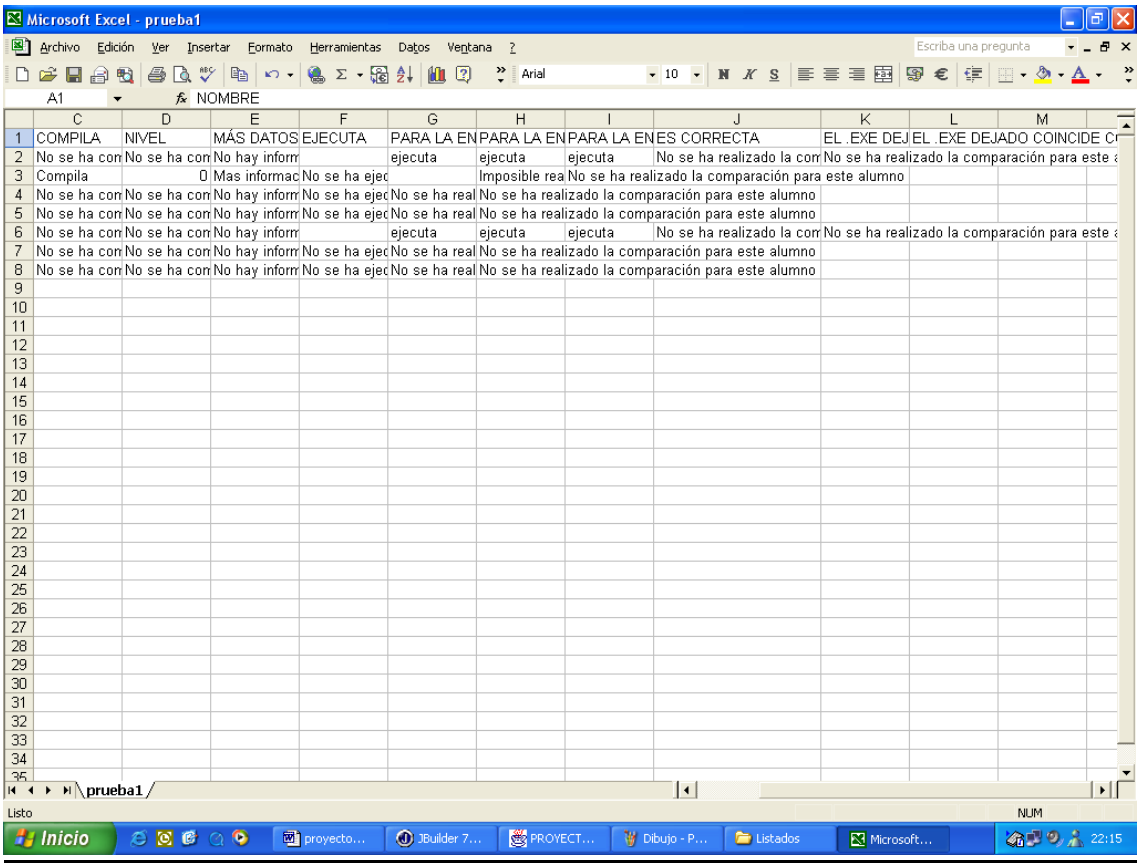

Pulsamos aceptar en la ventana de error y pulsamos de nuevo el botón *Generar Listado*.




Si pulsamos ahora sobre aceptar veremos que el nombre *prueba1* no existe y se generará el listado deseado.




Nos aparecerá este mensaje para indicar que no hubo ningún problema al escribir el listado. Ahora para ver el listado no tenemos más que ir al directorio actual de trabajo y entrar en la carpeta Listados. Dentro de esta carpeta encontraremos el listado de nombre *prueba1*. Éste lo podemos abrir con el Excel.




El archivo en Excel quedaría de esta forma que vemos más arriba.

## 4.2.4.2.- Generación de estadísticas

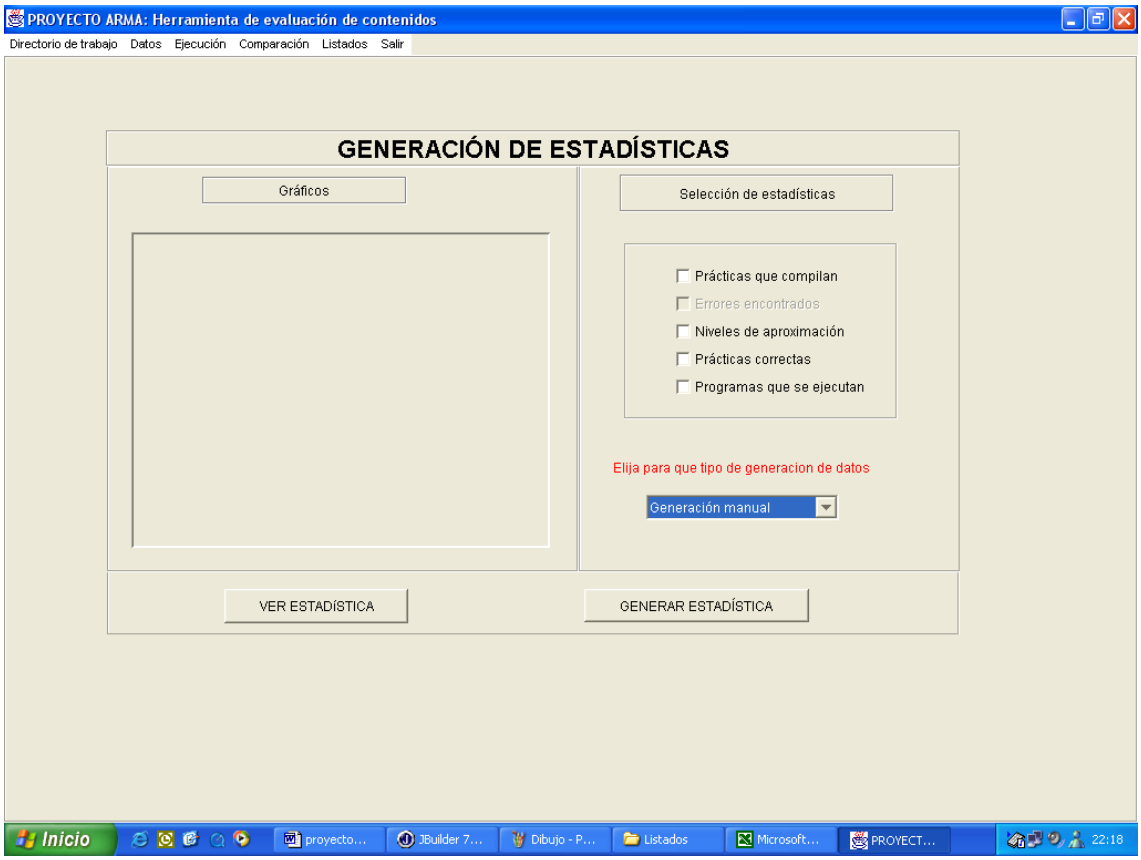
Si en vez de seleccionar en el menú principal la opción *Generar Listados* seleccionamos la opción *Generar Estadísticas*, llegaremos a la siguiente pantalla:

En esta pantalla podremos ver y generar estadísticas sobre los datos obtenidos sobre las prácticas de los alumnos.

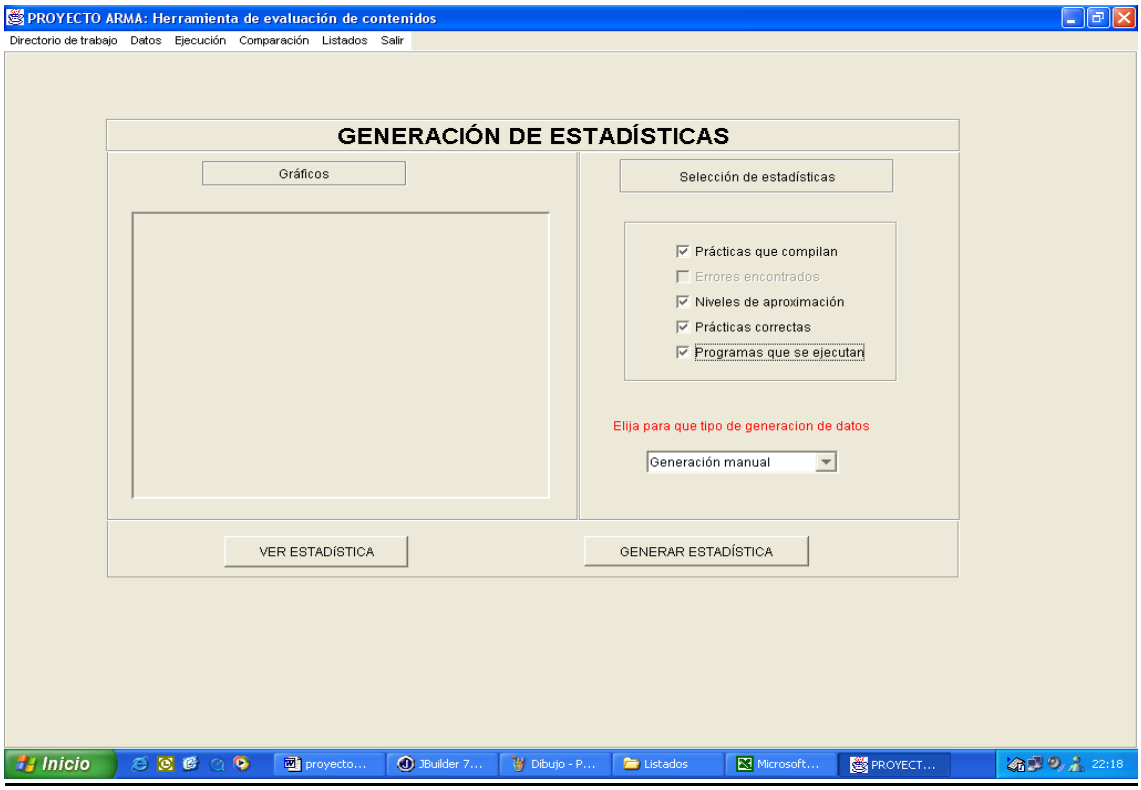
En el panel de la parte izquierda de la pantalla aparecerán las estadísticas si elegimos verlas. En la parte derecha de la pantalla están las estadísticas disponibles, que son las mismas opciones que para los listados. Se pueden elegir las opciones que queramos. En esta pantalla tendremos que elegir también, como en la pantalla de generación de listados, el tipo de generación de datos sobre el que se han hecho las pruebas sobre las prácticas de los alumnos.

### *4.2.4.2.a.- Ver una estadística por pantalla*

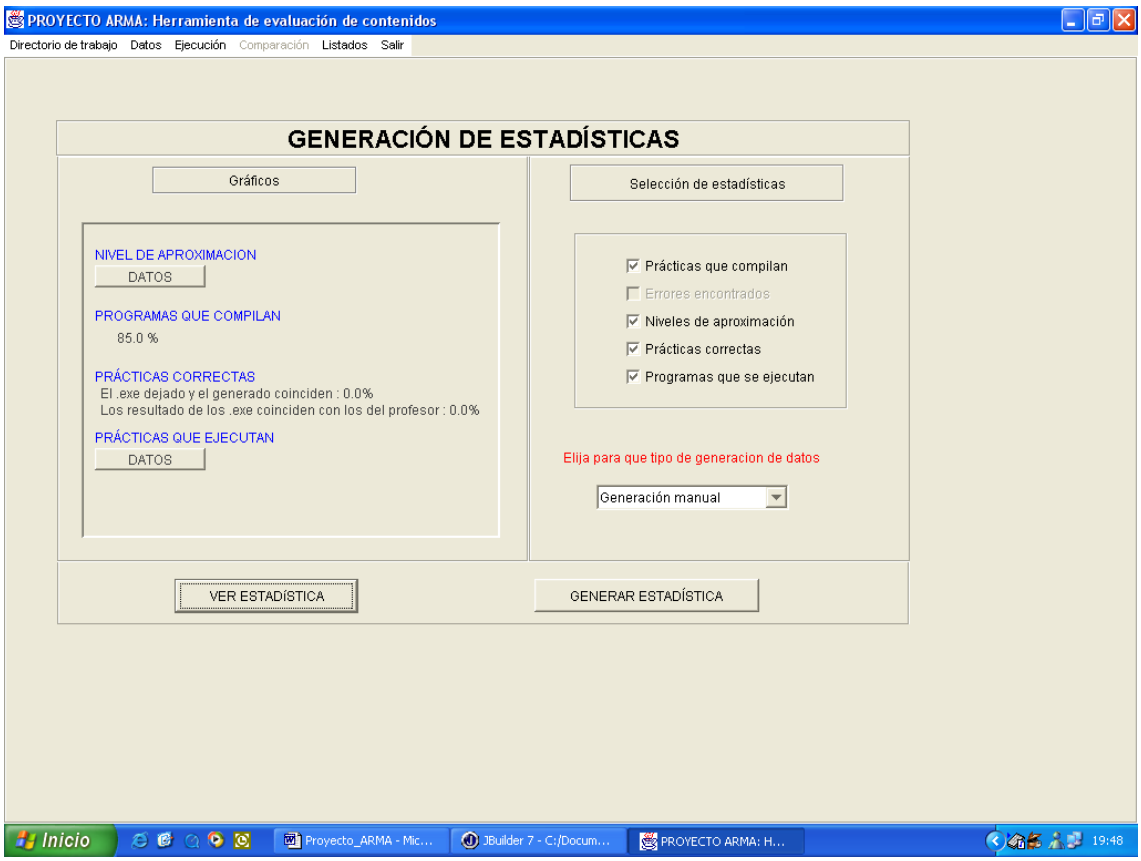
Si queremos ver las estadísticas, procederemos de la siguiente manera. Primero debemos seleccionar el tipo de generación de datos. En este ejemplo hemos elegido la generación manual de datos.




Una vez elegido el tipo de generación de datos, tendremos que elegir el tipo de estadística que queremos visualizar. Para este ejemplo hemos elegido ver el porcentaje de prácticas que compilan, que ejecutan, que son correctas y los distintos niveles de aproximación al código del profesor que hemos calculado para las distintas prácticas de los alumnos.




Ahora que ya tenemos el tipo de generación de datos y las estadísticas que queremos ver, pulsaremos sobre el botón *Ver Estadística*. Los datos aparecerán en el panel a la izquierda de la pantalla.



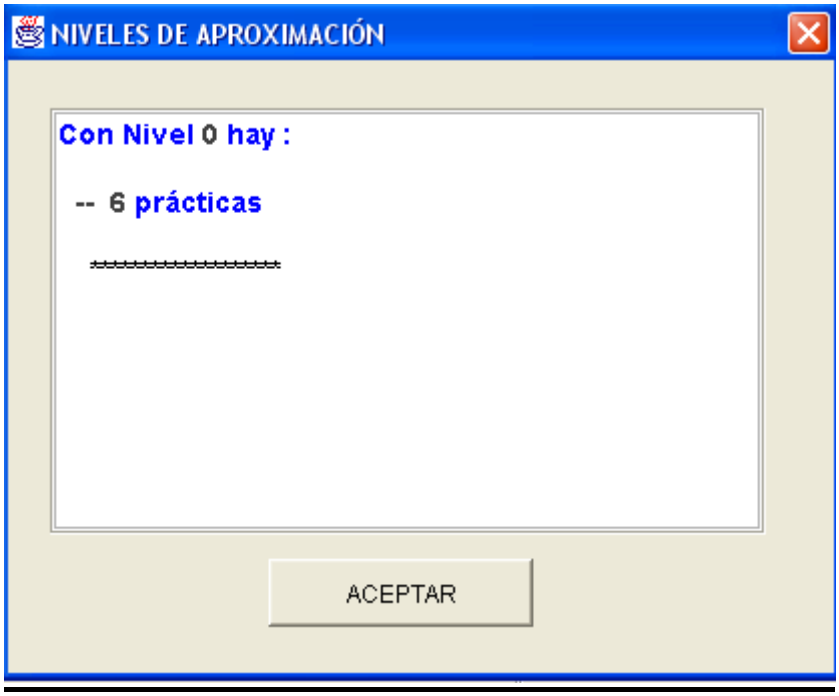
Como vemos, han aparecido a la izquierda los datos. Para los programas que compilan aparece directamente el porcentaje, ya que sólo se compilan una vez. En este caso compilan el 87% de las prácticas de los alumnos.

Para el porcentaje de prácticas correctas hemos tomado dos puntos de vista al respecto:

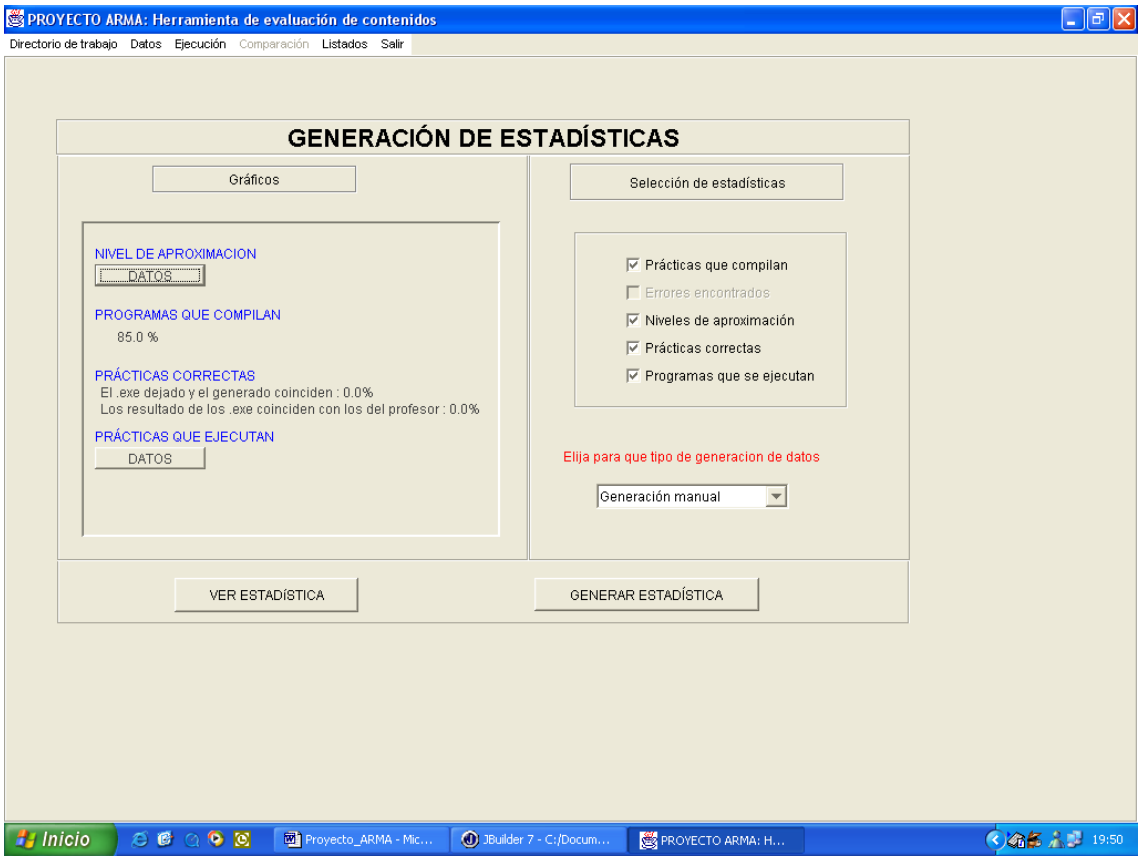
- Que la ejecución del *.exe* dejado por el alumno coincida con la ejecución del *.exe* generado por el *.pas* dejado también por el alumno. Las prácticas que no tienen ejecutable asociado al *.pas*, no se toman como correctas, sino como incorrectas.

- Que la ejecución del *.exe* dejado por el alumno coincida con la ejecución del *.exe* del profesor. Las prácticas que no tengan .pas asociado se toman como incorrectas.

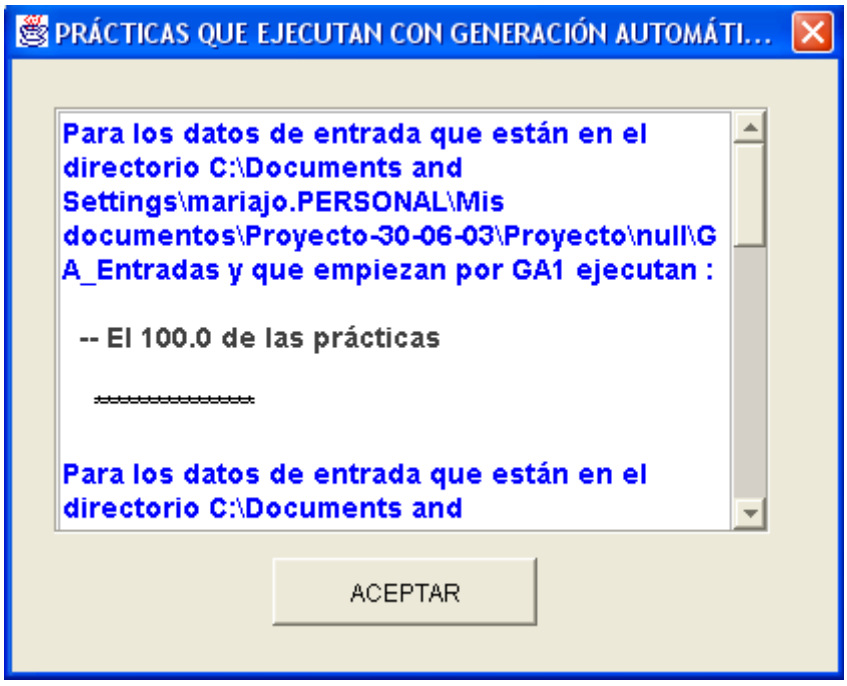
Para ver los distintos niveles de aproximación deberemos pulsar sobre el botón *Datos* debajo de la etiqueta *Niveles de aproximación*. Aparecerá otra ventana con la información sobre los niveles de aproximación.




En este caso vemos que de las seis prácticas que tenemos, todas tienen nivel 0 de aproximación al código del profesor. Si hubiera más niveles aparecerían igualmente en esta ventana. Cada nivel con el número de prácticas que lo tienen.



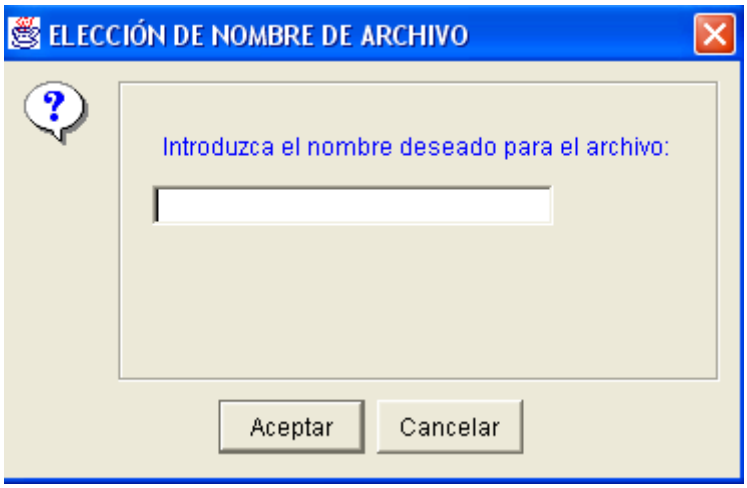

Para ver la información sobre el porcentaje de prácticas que se ejecutan deberemos pulsar sobre el botón *Datos* debajo de la etiqueta *Prácticas que ejecutan*. Al pulsar al botón aparecerá otra ventana con la información sobre las ejecuciones de los programas, para los distintos datos de entrada.



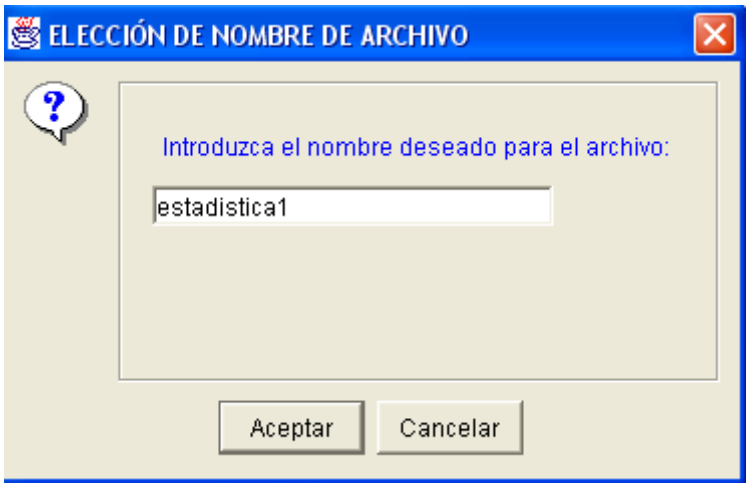
Como vemos aparece la información sobre el porcentaje de prácticas que se ejecutan en cada ejecución.

#### *4.2.4.2.b.- Generar un listado de estadísticas*

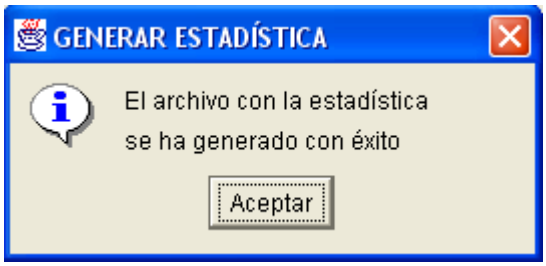
Si queremos generar un listado de estadísticas deberemos proceder como en el caso anterior, pero esta vez pulsar sobre el botón *Generar Estadística*. Al pulsar sobre el botón aparecerá un diálogo para introducir el nombre deseado para el listado de estadísticas.



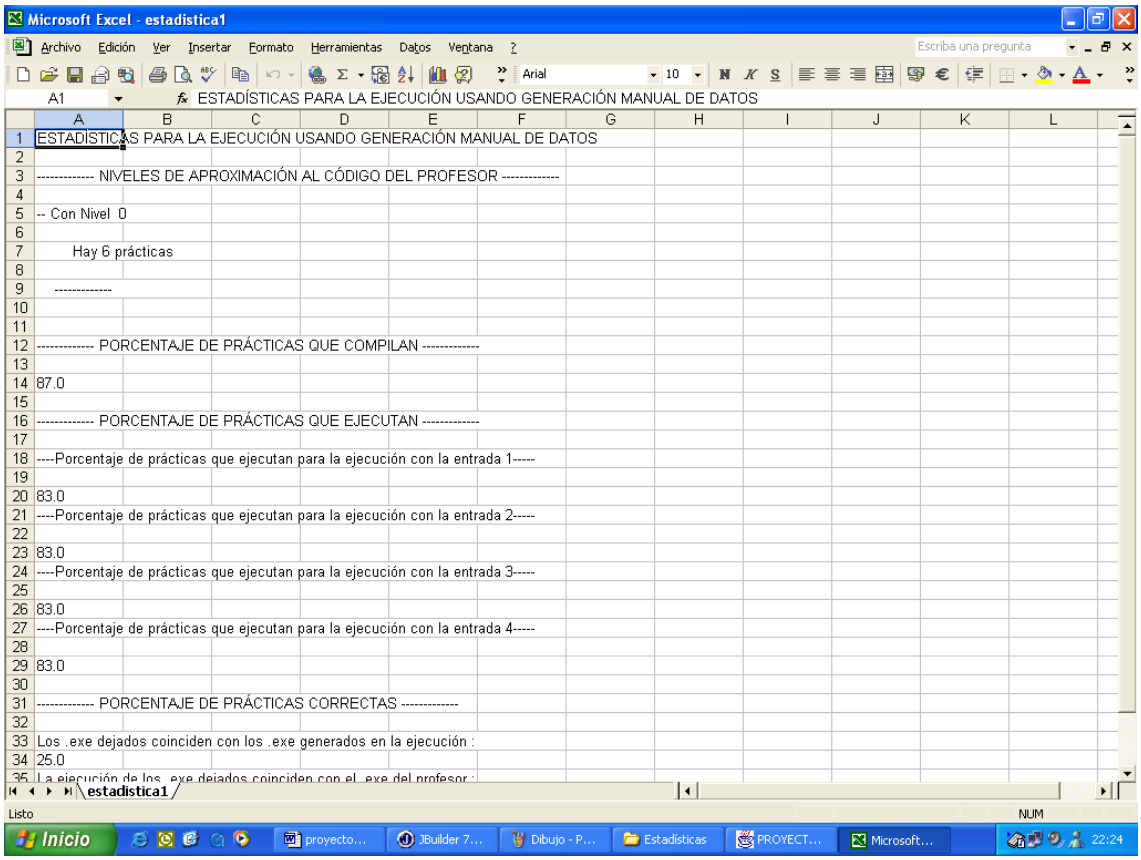

Introducimos el nombre deseado en el campo de texto. Como en el caso de los listados, si ya existe otro listado de estadísticas con ese nombre, el sistema nos informará de ello y no generará un listado de estadísticas. Los listados de estadísticas generados se guardarán en una carpeta llamada *Estadísticas* que se encontrará en el directorio de trabajo actual.




Hemos introducido el nombre *estadistica1*, si el nombre ya existiera, como hemos dicho antes, el lsitado de estadísticas no se crearía; en caso contrario el sistema nos informará del éxito de la operación.




En este caso el nombre era nuevo, así que se generó con éxito la estadística. Para ver el archivo generado, vamos al directorio actual de trabajo donde se encontrará una carpeta *Estadísticas*. Dentro de esta carpeta encontraremos el archivo *estadistica1* que podremos abrir con Excel y visualizar su contenido.



# 5.- CODIFICACIÓN DE LA HERRAMIENTA

## 5.1.- ENVÍO DE PRÁCTICAS A TRAVÉS DE INTERNET

Una vez el alumno ha introducido los datos y ha pulsado el botón de enviar (fichero paginaprincipal.html), se pasa a comprobar tanto la validez de los datos como el que estén todos los campos obligatorios rellenos. Esta comprobación se lleva a cabo en el archivo resultado.jsp En primer lugar se comprueba que se haya rellenado todos los campos, y a continuación se valida el nif del alumno, teniendo en cuenta que la longitud máxima del nif puede ser de 10 caracteres incluida la letra. Una vez se confronta con la lista alumnos y se encuentra el nif del alumno introducido (la búsqueda realizada es lineal dado que el número de datos es pequeño), se pasa a comparar los nombres. Para ello se extrae de la lista el nombre y los dos apellidos y se compara carácter a carácter con los introducidos. En caso de encontrar alguna diferencia, se crea un archivo de advertencia al profesor en el directorio: /upload/warnings/ indicando que se ha encontrado el alumno con el nif introducido, pero que se han observado diferencias entre los nombres, para que él compruebe luego si la diferencia es por alguna errata o si se trata de alumnos distintos y tomar las acciones adecuadas. En caso de que el nif introducido no concuerde con ninguno de la lista de alumnos, se genera un mensaje de error. También se genera un mensaje de error en el caso de que la práctica no sea un número y se haya introducido cualquier otro tipo de dato.

A la hora de subir los archivos de los alumnos, se comprueba que el nombre del fichero tiene el formato correcto y si es así se subirá al servidor en el directorio: /upload/practica*X*/ siendo X el número de práctica al que corresponden los ficheros que se quieren subir al servidor. En caso contrario se genera un mensaje de error. Esta comprobación se realiza en el fichero paso2.jsp

## 5.2.- HERRAMIENTA DE EVALUACIÓN DE CONTENIDOS

En este apartado se explicarán algunas de las cuestiones más relevantes de la aplicación Java *(herramienta de evaluación de contenidos)*, para aquellas clases que lo requieran.

Primero de todo decir que la aplicación está diseñada para la utilización de prácticas escritas en Pascal (7.0), que leen datos de ficheros de texto, y del mismo modo escriben los resultados en ficheros de texto. La única restricción impuesta en principio es que las prácticas terminen, es decir puede que ejecuten o no, pero si ejecutan deben terminar.

## 5.2.1.- Generación de datos

### 5.2.1.1.- Clase *PanelGenAutomatica()*

La clase recoge las características de los datos que se deben generar, comprobando previamente si son correctos (comprobando si se corresponden con el tipo especificado y si el rango de valores es correcto).

Conviene destacar en este caso, que los tipos de datos que la aplicación puede generar, son los tipos de datos simples:

- caracteres (**char**),
- booleanos (**boolean**),
- enteros (**integer**),
- y reales (**real** en *formato estándar*, y **real** en *formato científico*).

Para ello se utilizan los siguientes métodos:

```
compruebaBool()
compruebaChar()
compruebaInt()
compruebaReal()
compruebaRealC()
```

A continuación se generan los datos utilizando los siguientes métodos:

```
generaBoolAleat()
generaCharAleat()
generaIntAleat()
generaDobEstAleat()
generaDobCientAleat()
```

Conviene destacar como se realiza la generación de números reales en formato científico, ya que la idea era generar este tipo de datos con el formato que haya introducido el usuario (para los valores mínimo y máximo del rango). Para poder guardar la información relativa a este formato se utiliza una clase **`FormatCientifico()`**, con los siguientes atributos privados:

```
public class FormatCientifico {

  private boolean tieneE;   /* Guarda si el número contiene el
                               carácter de exponente 'E' (indica
                               si el número tiene exponente o no) */
  private boolean tieneSigno;   /* Guarda si el número tiene
                                   signo o no */
  private char signo;           /* Guarda el signo del número:
                                   '+', o '-' */
  private int exponente;        /* Guarda el valor del exponente
                                   del número */

.
.
.
```

Esta clase contiene además los métodos necesarios para acceder y actualizar dichos atributos.

Mediante esta clase podemos saber el formato que introdujo el usuario (si introdujo dos formatos distintos para el valor mínimo y el máximo, *se coge por defecto el del valor mínimo*).

Así pues después de saber si el formato introducido es correcto (recordemos, el formato científico para el tipo *real* en Pascal de forma genérica es (+/-) parteEntera(.parteDecimal)(E (+/-) Exponente) [los paréntesis indican que lo que va entre ellos puede o no aparecer]) la aplicación pasa a generar un valor real aleatorio entre ese rango y con el formato especificado.

Dado que Java tiene su propio formato científico, cuando se generaba un número demasiado grande o pequeño, la herramienta lo transformaba a su propio formato. Para solucionarlo se han diseñado métodos que comprueban si esta transformación había ocurrido, y en tal caso deshacer los cambios para guardar los valores en el formato Pascal correspondiente. Además para crear los valores dentro del rango especificado ha sido necesario, primero transformar tanto el valor mínimo y el máximo, en un mismo formato (en este caso real estándar), generar el valor, y después volver a transformarlo de manera manual al formato científico especificado por el usuario.

La generación de números reales en formato estándar mantiene *el mismo formato, en cuanto a la posición del punto decimal, que el que había introducido el usuario* (al igual que antes si existen diferentes formatos para el valor mínimo y el máximo del rango, *se coge por defecto el del valor mínimo*). Para ello se guarda la información acerca de la posición del punto decimal, y se descompone el número real, en dos enteros:

- uno correspondiente a la parte entera
- y otro correspondiente a la parte decimal

Con esto, se generan valores aleatorios enteros para las dos partes (siempre dentro de los rangos permitidos), y después se genera el número real en formato estándar resultado de componer ambas partes y de poner el punto decimal en el sitio correspondiente

La clase **`PanelGenAutomica()`**, hace uso de la otra clase **`DatFich()`**, que guarda la información para cada fichero que se quiere generar. Esta información que guarda viene dada por los atributos de la clase, que son los siguientes:

```
public class DatFich {

  private String nombre;     /* Nombre del fichero */
  private String tipo;       /* Tipo del fichero (de texto o
                                binario) */
  private int numValores;    /* Numero de datos que contendrá el
                                fichero */
  private ValoresFich [] arrayVal;   /* Valores de los datos del
                                        fichero */


      .
.
      .
```

Además contiene los métodos para actualizar y devolver los atributos correspondientes. Se puede observar que hace referencia a otra clase **`ValoresFich()`**, que guarda información acerca de los valores que se deberán generar posteriormente para un fichero determinado:

```
public class ValoresFich {

  private String tipo;   /* Guarda el tipo del dato */
  private String minimo;  /* Guarda el valor mínimo del dato */
  private String maximo;  /* Guarda el valor máximo del dato */
  private boolean hayFinLinea; /* Indica si hay fin de línea
                                  después del dato */

      .
.     .
```

## 5.2.1.2.- Clase *PanelGenManual()*

Esta clase permite al usuario seleccionar una serie de ficheros de entrada para cada una de las generaciones especificadas. La clase hace uso de otra clase **`PFormatoManual()`** que será la encargada de gestionar la lista de ficheros adjuntados, permitiendo al usuario renombrar los ficheros, o seleccionar alguno nuevo (estas opciones se presentan como un menú que aparece al pinchar con el botón derecho del ratón sobre el nombre del fichero seleccionado, se utiliza un objeto de tipo **`JPopupMenU()`**, y el evento al que responde esta acción de presentar dicho menú es ***`objeto`*`_moussePressed (Event evento)`**. Se utiliza la siguiente condición:

```
if (((e.getModifiers())&(e.BUTTON3_MASK))!=0)
```

para comprobar que el botón pulsado ha sido el derecho.)

Conviene destacar que para que la aplicación controle que el usuario no selecciona cualquier tipo de fichero (sino solo ficheros de texto), esta clase **`PFormatoManual()`**, hace uso de otra clase: **`FiltroTxt()`**, es una clase que extiende de **`FileFilter()`** y para la cual se redefine el método ***accept()***, de manera que en el seleccionador de ficheros que se le presenta al usuario (objeto **`JFileChooser()`**) solo se le presenten al usuario directorios y archivos de tipo *.txt*

```
public class FiltroTxt extends FileFilter {

  final static String exe = "txt";

  /*************************************************************/

  /** Metodo que devuelve cierto si dado un determinado archivo,
   *  tiene extension 'txt', y devuelve falso en otro caso
   */

  public boolean accept(File f) {

    if (f.isDirectory()) {
      return true;
    }

    String s = f.getName();
    int i = s.lastIndexOf('.');

    if (i > 0 &&  i < s.length() - 1) {
      String extension = s.substring(i+1).toLowerCase();
      if (exe.equals(extension))
        return true;
      else
        return false;
    }

    return false;
  }

  /*************************************************************/

  // Descriptor del filtro
  public String getDescription() {
    return "Ficheros de texto (.txt)";
  }

  /*************************************************************/

} /* fin de la clase FiltroTxt */
```

Una vez que el usuario ha seleccionado todos los ficheros (la aplicación comprueba que se rellenan todos los datos que inicialmente especificó el usuario), se hacen copias de esos ficheros, para que sigan estando en su ubicación actual, pero también se encuentren en la carpeta *dir_de_trabajo*/**GM_Entradas**, que es condición necesaria para que la aplicación pueda utilizar posteriormente dichos datos.

En ambos tipos de generaciones (automática y manual), se han implementado métodos que borran de manera recursiva todas las carpetas o ficheros relacionados con la generación respectiva (si es que existieran), cuando se desea hacer una nueva generación de datos. Conviene tener en cuenta que solo se borran si el usuario realiza una nueva generación de datos correcta; en caso contrario la aplicación sigue guardando los datos anteriores con los que trabajaba.

Así para la generación automática se definen:

```
void borrarCarpetasAntiguas(String nomDirectorio){

    //coger los ficheros del directorio
    File dir=new File(nomDirectorio);
    File[] listaFich = dir.listFiles();
    //si es un directorio hacer  un borrado recursivo
    for (int i=0; i<listaFich.length; i++){
      if(listaFich[i].isDirectory()) {
        if (esCarpetaGA(listaFich[i].getName())) {
          borrarRecursivamente(listaFich[i]);
          listaFich[i].delete();
        }
        else
          if ((listaFich[i].getName()).equals("Compiladas"))

borrarCarpetasAntiguas(directorio+"\\Compiladas\\ExeGenerados");
        else
          if ((listaFich[i].getName().equals("CompConProf"))) {
        borrarCarpetasAntiguas(directorio+"\\CompConProf");
        if (
((new File(directorio+"\\CompConProf")).listFiles()).length == 0)
          listaFich[i].delete();
          }
      }
      else
        if
((listaFich[i].getName().compareTo("ListaEjecutadasAutomatica.txt")==0)
           ||(esCarpetaGA(listaFich[i].getName())))
          listaFich[i].delete();
    }
  }

  /*************************************************************/


  void borrarRecursivamente(File dir) {

    File[] listaFich = dir.listFiles();
    for (int i=0; i<listaFich.length; i++){
      if(listaFich[i].isDirectory()) {
        borrarRecursivamente(listaFich[i]);
        listaFich[i].delete();
      }
      else
        listaFich[i].delete();
    }
  }

  /*************************************************************/
```

y respectivamente para la generación manual se definen:

```
void borrarCarpetasAntiguas(String nomDirectorio){

    //coger los ficheros del directorio
    File dir=new File(nomDirectorio);
    File[] listaFich = dir.listFiles();
    //si es un directorio hacer  un borrado recursivo
    for (int i=0; i<listaFich.length; i++){
      if(listaFich[i].isDirectory()) {
        if (esCarpetaGM(listaFich[i].getName())) {
          borrarRecursivamente(listaFich[i]);
          listaFich[i].delete();
        }
        else
          if ((listaFich[i].getName()).equals("Compiladas"))

borrarCarpetasAntiguas(directorio+"\\Compiladas\\ExeGenerados");
        else
          if ((listaFich[i].getName().equals("CompConProf"))) {
        borrarCarpetasAntiguas(directorio+"\\CompConProf");
        if (
((new File(directorio+"\\CompConProf")).listFiles()).length == 0)
          listaFich[i].delete();
          }
          else
            if ((listaFich[i].getName().equals("DatosEspecificos"))) {
          borrarRecursivamente(listaFich[i]);
          listaFich[i].delete();
            }
      }
      else
        if
((listaFich[i].getName().compareTo("ListaEjecutadasManual.txt")==0)
            ||(esCarpetaGM(listaFich[i].getName())))
          listaFich[i].delete();
    }
  }

  /*************************************************************/

void borrarRecursivamente(File dir) {

    File[] listaFich = dir.listFiles();
    for (int i=0; i<listaFich.length; i++){
      if(listaFich[i].isDirectory()) {
        borrarRecursivamente(listaFich[i]);
        listaFich[i].delete();
      }
      else
        listaFich[i].delete();

    }

  }
```

## 5.2.2.- Ejecución de programas

### 5.2.2.1.- Clase *PanelEjecucion()*

Es la principal de la aplicación, ya que en ella se realizan la mayoría de las funcionalidades especificadas para el sistema, así como la generación de toda la información que es necesario ir almacenando. Esta clase, además de controlar en todo momento si existen datos para cada una de las generaciones, se encarga de *compilar, ejecutar los programas* (según las diferentes opciones que se le presentan al usuario), *generar las salidas y carpetas con la información oportuna, realizar comparaciones cuando sea necesario así como generar todos los listados y ficheros de texto con la información específica para cada tipo de ejecución y/o comparación*. En su implementación cabe destacar:

- **Llamada al compilador de Pascal**:

El usuario que utilice la aplicación tendrá el compilador de Pascal en su PC (por defecto se suele instalar en el directorio **c:/tp/bin** con el nombre **tpc.exe**; además esta es la ruta que utiliza la aplicación para llamar a dicho programa).

El objeto **`Runtime()`** de Java, tiene un método ***exec***, que permite ejecutar la aplicación que se le pase como parámetro, así como diferentes parámetros de entrada que se le pueden enviar también. El programa ***tpc.exe*** se ejecuta desde el terminal *ms-dos* con la siguiente orden: **> tpc *nombre_práctica***. Lo que necesitamos es lanzar en nuestra aplicación una orden *ms-dos* y no un programa concreto. Esto es *posible realizarlo mediante la creación un archivo por lotes (.bat)*, que tenga la orden 'tpc nombre_practica. Para hacer esto implementamos un método llamado ***generaBat()***.

Se redirecciona la salida a un fichero de texto (ya que es la propia aplicación la que se encarga de gestionar estos resultados, para poder generar luego los listados y los ficheros de información oportunos).

```
public void generaBat(String nombreBat, String lineaComando) {
   try {
     BufferedWriter program= new BufferedWriter (new
                             FileWriter(nombreBat));
     program.write(lineaComando);
     program.close();
     Process p = Runtime.getRuntime().exec(nombreBat);
     p.waitFor();
     /** BORRAMOS EL .BAT GENERADO */
     File borraProgram = new File(nombreBat);
     borraProgram.delete();
   }
   catch(Exception e) {
     System.out.println("excepcion "+e);
   System.out.println("Se produjo una excepcion");}

 }
```

Se puede observar que los *parámetros* de este método son el nombre (con la ruta), del fichero *.bat* que se va a crear, y la orden *ms-dos* (línea de comando), que contendrá dicho fichero. Se observa también que después de utilizarlo se borra puesto que son archivos que no tienen por qué guardarse. Por lo tanto la instrucción

```
Process p = Runtime.getRuntime().exec(nombreBat);
```

ejecuta este archivo *.bat*, en lugar de una aplicación *.exe*.

La línea de comando que se le pasa como parámetro es la siguiente:

***ruta_tpc  nombre_ practica***

donde *ruta_tpc* sería la ruta y el nombre (sin extensión) donde esta ubicado el programa **tpc.exe**, por ejemplo **c:/tp/bin/tpc**, y el *nombre_practica* es el nombre de la práctica del alumno sin extensión (sin *.pas*).

(En este caso si se genera un ejecutable para la práctica entonces es que compila, y en cualquier otro caso, es que la práctica no compila). Se comprueba con las siguientes líneas de código:

```
File ejecutable=new File(nombreEjecutable);
boolean existe=ejecutable.exists();
```

donde *nombreEjecutable*, es el nombre de la practica, con extensión *.exe*

- **Ejecución de las prácticas**:

Para el caso de ejecutar las prácticas la línea de comando que se le pasa como parámetro al método ***generaBat()*** es la siguiente:

***directorio_practica/nombrePractica >> fichero_donde_se_redirecciona_la_salida***

Donde *directorio_practica* es el directorio donde se encuentra el ejecutable y el *nombrePractica*, es el *.exe* que se quiere ejecutar. En este caso es necesario redireccionar la salida a un fichero de texto, que estará vacío si se ejecutó la práctica de forma correcta, o bien no existirá o contendrá el error de ejecución correspondiente si la práctica no se ejecutó de forma satisfactoria. Comprobando este fichero, es por lo tanto como comprobamos si la practica ejecuta o no ejecuta. El código que comprueba esto en la aplicación es el siguiente:

```
generaBat((directorio+"\\"+"myprog.bat"),
          (directorio+"\\"+fich.getName()+">>"+nombreSalEj));
if (!(new File(nombreSalEj).exists()))
  ejecuta = false;
else
  ejecuta = hayError(nombreSalEj);

/** Borramos los ficheros auxiliares que hemos utilizado **/
File borraErrEj = new File(nombreSalEj);
borraErrEj.delete();
```

la función ***hayError()*** está implementada de la siguiente forma:

```
public boolean hayError (String nombreSalEj) {
  try {
    boolean ej=true;
    FileReader errEj = new FileReader(nombreSalEj);
    StreamTokenizer conjLex = new StreamTokenizer(errEj);
    int token1=conjLex.nextToken();
    while (token1!=conjLex.TT_EOF) {
      if (token1==conjLex.TT_WORD)
        if ((conjLex.sval).equals("Runtime")) {
      int token2=conjLex.nextToken();
      if (token2==conjLex.TT_WORD)
        if((conjLex.sval).equals("error")) {
      ej = false;
      break;
        }
        }
        token1=conjLex.nextToken();
    }
    errEj.close();
    return ej;
  }
  catch (Exception e) {System.out.println(e.getMessage());
  return false; }
}
```

- **Ejecución de los ficheros *.bat*:**

1.- Los nombres de las practicas deben tener formato MS-DOS, esto es, no podían contener más de ocho caracteres, ya que la orden dada en el fichero .bat es una orden MS-DOS. En nuestro caso, el formato para el nombre de las prácticas es **P*X*D*Y***, siendo ***X*** el número de practica e ***Y*** el *dni* del alumno, de esta forma los nombres superan esa longitud permitida. Por lo tanto se ha implementado métodos para renombrar las prácticas, utilizarlas, y posteriormente volver a ponerles su nombre original.

2.- Como la aplicación esta diseñada para prácticas que leen y escriben en ficheros de texto, se llegó a la determinación que los '*assign*', para estos ficheros dentro del código en Pascal, no debían llevar rutas, de manera que las prácticas leerían/escribirían de/en los ficheros de texto que se encontraran en el directorio por defecto. Según esto, el directorio por defecto que toman las prácticas es el directorio desde el que se ejecuta la aplicación Java (puesto que es la aplicación principal que está en todo momento en curso, y por lo tanto desde la cual se ejecuta el archivo por lotes *.bat* que contiene la línea que ejecuta las prácticas, es decir, el programa que ejecuta realmente las prácticas es la aplicación Java, de manera que es el directorio de esta el que toman por defecto los programas que se ejecuten desde el *.bat*). Por un lado según esto, las salidas que generará cada practica lo hará en el directorio donde se ejecuta la aplicación Java (nuestra aplicación de herramienta de evaluación de contenido), lo cual no es deseable, primero porque las salidas se irían sobrescribiendo, y porque además no tiene sentido guardarlas ahí. Así pues, se tuvieron que implementar métodos también que movieran estas salidas, desde el directorio de la aplicación Java, hasta su lugar correspondiente que sería: ***directorio_de_trabajo/tipoGen*_Salidas** (siendo *tipoGen* automática (GA) o manual (GM), según el tipo de generación seleccionada por el usuario.)

Para el caso de las entradas, ocurre justamente los contrario, las entradas se encuentran en ***directorio_de_trabajo/tipoGen*_Entradas** (siendo *tipoGen* automática (GA) o manual (GM), según el tipo de generación seleccionada por el usuario.). Según lo comentado anteriormente es necesario que se encuentren en el directorio desde el que se ejecuta la aplicación Java antes de que se ejecuten las prácticas de los alumnos. Para esto, entonces se han implementado también métodos que primero mueven las entradas de ***directorio_de_trabajo/tipoGen*_Entradas** al directorio desde donde se ejecuta la aplicación Java; después se realiza la operación oportuna solicitada por el usuario (por ejemplo ejecución de los *.exe* de los alumnos), y una vez realizada la acción se vuelven a mover las entradas a su directorio original, es decir, se mueven del directorio desde donde se ejecuta la aplicación Java a ***directorio_de_trabajo/tipoGen*_Entradas**. Se hace para cada una de las generaciones que seleccionó el usuario, por eso se utiliza un recorrido ***for***; en el programa la parte correspondiente a esto, por ejemplo para la opción de ejecución de los *.exe* de los alumnos es la siguiente:

```
for(int i=1;i<=numEjec;i++) {
  String [] nomEnt=tratarEntrada(i,esManual);
  ejecutaProgramas(dir,i,esManual,nomEnt,bufferEscritura);
  removerEntrada(i,nomEnt,esManual);
}
```

donde ***tratarEntrada()*** se implementa como sigue:

```
public String [] tratarEntrada(int i,boolean esManual) {
  String [] aux=null;
  int indArray=0;
  try {
    if (!esManual) {
      File dirEntradas = new File(dirActual+"\\GA_Entradas\\");
      File [] arrayEntradas = dirEntradas.listFiles();
      aux = new String[arrayEntradas.length];
      for (int j=0; j<arrayEntradas.length; j++) {
        String nombre = arrayEntradas[j].getName();
        char[] caracteres = nombre.toCharArray();
        int indice = 2;
        String numero = "";
        while (caracteres[indice]!='_') {
          numero = numero+caracteres[indice];
          indice++;
        }
        indice++;
        String nomFich="";
        for (int m=indice; m<caracteres.length; m++)
          nomFich = nomFich+caracteres[m];
        int numEnt = (new Integer(numero)).intValue();
        if (numEnt == i) {
          File nuevoFich = new File(nomFich);
          arrayEntradas[j].renameTo(nuevoFich);
          aux[indArray] = nomFich;
          indArray++;
        }
      }
    }
    else{  // Es generacion manual
      File dirEntradas=new File(dirActual+"\\GM_Entradas\\GM_Entrada"+i+"\\");
      File [] arrayEntradas = dirEntradas.listFiles();
      aux = new String[arrayEntradas.length];
      for (int j=0; j<arrayEntradas.length; j++) {
        String nomFich = arrayEntradas[j].getName();
        File nuevoFich = new File(nomFich);
        arrayEntradas[j].renameTo(nuevoFich);
        aux [indArray]= nomFich;
        indArray++;
      }
    }
  }
  catch (Exception e) {
  System.out.println("Se produjo una excepción en TRATAR ENTRADA");}
  return aux; }
```

y ***removerEntrada()***:

```
public void removerEntrada(int i,String [] cadena,boolean esManual) {

  try {
    File dirEntradas = new File(".\\");
    File [] arrayEntradas = dirEntradas.listFiles();
    for (int j=0; j<arrayEntradas.length; j++) {
      String nombre = arrayEntradas[j].getName();
      for (int a=0;a<cadena.length;a++) {
        if (cadena[a] == null)
          break;
        else {
          if (cadena[a].equals(nombre)) {
            if (!esManual) {
              File nuevoFich = new File(dirActual+"\\GA_Entradas\\GA"+i+
                  "_"+cadena[a]);
              arrayEntradas[j].renameTo(nuevoFich);
              break;
            }
            else {
              File nuevoFich = new File(dirActual+"\\GM_Entradas\\GM_Entrada"+
                  i+"\\"+cadena[a]);
              arrayEntradas[j].renameTo(nuevoFich);
              break;
            }
          }
        }
      }
    }
  }
  catch (Exception e) {
  System.out.println("Se produjo una excepción en remover entrada");}
}
```

## 5.2.3.- Comparación de las estructuras de los programas

### 5.2.3.1.- Clase *PanelComparacion()*

Esta clase gestiona todo lo relacionado con la comparación de las estructuras de los programas de los alumnos con la estructura del programa del profesor (que siempre debe encontrarse en el directorio de trabajo y cuyo nombre tiene como formato estándar **P*X*PROF**, siendo ***X*** el número de practica). Como ya se ha dicho anteriormente la comparación se realiza para las prácticas que compilan (pues en otro caso no tiene sentido). Los pasos a seguir son:

**5.2.3.1.a.- Normalización** de dichas prácticas (supone estandarizar su formato para facilitar la posterior comparación)
**5.2.3.1.b.- Comparación 1** (realiza una comparación superficial sobre el número de variables, instrucciones de control, de operadores, subprogramas definidos y de líneas de código).
**5.2.3.1.c.- Comparación 2** (realiza una comparación más detallada, basándose realmente en la estructura de los programas)

El *nivel de aproximación* de las prácticas con las del profesor, se mide mediante un número de forma que *cuánto mayor sea este número menor es la aproximación*, ya que a este número se le van sumando puntos por cada parte con la que no coincide en la práctica del profesor.

#### *5.2.3.1.a.- Normalización*

Para la comparación de dos programas se realiza una normalización previa, para que todos los programas tengan el mismo formato, y resulte más fácil su comparación. Se decidió la estructura de los programas normalizada siguiente:

- Los programas normalizados no deben de tener comentarios, durante el proceso de normalización se ignoran los comentarios en Pascal, de los dos tipo, encerrados entre {} y entre (* *). Se ignoran también las cadenas de caracteres constantes, por ejemplo, en las instrucciones de escritura por pantalla. Estas cadenas se ignoran, ya que no son relevantes a la hora de comparar la estructura de dos programas.

- En los programas normalizados no debe haber líneas en blanco entre las instrucciones, en el proceso de normalización se suprime cualquier línea en blanco, y los espacios blancos sobrantes.

- Cada instrucción debe de aparecer en líneas distintas.

- El begin y el end deben de estar también solos en una línea cada uno.

- Las instrucciones de control aparecen igualmente solas en una línea.

- El programa normalizado estará escrito todo en letras minúsculas.

- El programa normalizado debe de ser correcto, es decir, debe seguir compilando en Pascal.

- Para normalizar un programa, es necesario que éste compile. Si no compila el programa entregado por el alumno no tiene sentido compararlo con el programa del profesor u otros programas, ya que la práctica del alumno no sería correcta desde un principio.

Para entender mejor el proceso de normalización, podemos verlo con un ejemplo de un programa en Pascal sin normalizar, y el resultado de dicha normalización:

```
{21 Enero 2003}

PROGRAM CalculoMCD (input, Entrada,output,Salida);

VAR
  Numero1, Numero2: Integer;
  Entrada,Salida: Text;

{RECURSIVIDAD}
FUNCTION MCD(Num1, Num2: Integer):Integer;
  BEGIN
  IF Num1>Num2 THEN MCD:=MCD(Num1-Num2, Num2)
  ELSE IF Num1 < Num2 THEN MCD:= MCD(Num1, Num2 - Num1)
  ELSE MCD:= Num1
  END;

BEGIN
```

```
  Assign(Entrada,'Entrada.txt');
  Assign(Salida,'Salida.txt');
  Reset(Entrada);
  Rewrite(Salida);
  Readln(Entrada,Numero1);
  Readln(Entrada, Numero2);
  Writeln(Salida,'m.c.d(', Numero1,',',Numero2,') = ',MCD(Numero1, Numero2));
  Close(Entrada);
  Close(Salida);
END.

{BUCLE}
{WHILE Numero1 <> Numero2 DO
    BEGIN
    IF Numero1> Numero2 THEN Numero1:= Numero1 - Numero2
    ELSE Numero2:=Numero2 - Numero1
    END;
 Writeln(' El mcd es',Numero1);}
```

El programa anterior es un programa en Pascal que no ha sido todavía normalizado, como podemos ver tiene comentarios que no debería tener el programa normalizado. También escribe cadenas de caracteres constantes que se ignorarán al normalizarlo. También podemos ver que no está todo en minúsculas, al normalizarlo, quedará todo en letra minúscula.

```
program calculomcd(input,entrada,output,salida);
var
numero1,numero2:integer;
entrada,salida:text;
function mcd(num1,num2:integer):integer;
begin
if num1>num2 then
mcd:=mcd(num1-num2,num2)
else
if num1<num2 then
mcd:=mcd(num1,num2-num1)
else
mcd:=num1;
end;
begin
assign(entrada,'entrada.txt');
assign(salida,'salida.txt');
reset(entrada);
rewrite(salida);
readln(entrada,numero1);
readln(entrada,numero2);
writeln(salida,'',numero1,'',numero2,'',mcd(numero1,numero2));
close(entrada);
close(salida);
end.
```

Hemos normalizado el programa anterior, y este es el resultado. Como se puede observar, los comentarios y las cadenas constantes de caracteres se han ignorado. Cada instrucción va sola en una línea, los begin y end también cumplen los requisitos de un programa normalizado.

#### *5.2.3.1.b.- Comparación 1*

Este tipo de comparación debe guardar información acerca del **número de operadores**, de **variables**, de **subprogramas definidos**, de **líneas de código**, y de **instrucciones de control**, para las prácticas de los alumnos y para la del profesor y poder realizar así después la comparación entre ambos datos.

Esta comparación realiza los cálculos del nivel de aproximación según lo siguiente:

- La *diferencia máxima de número de variables* que puede tener el alumno declaradas en comparación con las que tiene el profesor es *2*.

- La *diferencia máxima de número de operadores* utilizados que puede tener el alumno en comparación con los que tiene el profesor es *2*.

- La *diferencia máxima de número de líneas de código* que puede tener el alumno en comparación con las que tiene el profesor es *2*.

- *No se permite diferencia entre el número de instrucciones de control* que tiene el alumno y el numero que tiene el profesor.

- *No se permite diferencia entre el número de subprogramas definidos* que tiene el alumno y el numero que tiene el profesor.

Para los casos en los que este permitida la diferencia, se utiliza un método implementado en la clase **`PanelComparacion()`** que es el siguiente:

```
public int permitidaDiferencia (int numero, int nivAprox) {
  switch (numero) {
    case 0: break;
    case 1: { nivAprox = nivAprox + 1; break; }
    case 2: { nivAprox = nivAprox + 2; break; }
    default: { nivAprox = nivAprox + 5; break; }
  }
  return nivAprox;
}
```

así, se suman al nivel tantos puntos como diferencias existan, pero si se supera la diferencia máxima permitida, entonces se suma siempre una constante que hemos prefijado con valor 5.

Por lo tanto para los casos en los que no esté permitida la diferencia, se suma siempre este valor 5, mediante el siguiente método implementado en la clase **`PanelComparacion()`**:

```
public int noPermitidaDiferencia (int numero, int nivAprox) {
  if (numero != 0)
    nivAprox = nivAprox + 5;
  return nivAprox;
}
```

Debemos tener, por tanto, una clase que guarde la información acerca del número de variables, operadores, líneas de código, instrucciones, y definición de subprogramas para cada práctica, dicha clase se ha llamado clase **`Comparacion1()`**. Los atributos de esta clase son:

```
/** atributos propios de la clase Comparacion1 **/
private int operAritmeticos; /** número de operadores aritméticos del
                                programa  principal **/
private int operLogicos; /** número de operadores lógicos del programa
                            principal **/
private int varBooleanasPrin; /** número de variables booleanas del programa
                                 principal **/
private int varEnterasPrin; /** número de variables enteras del programa
                               principal **/
private int varCaracterPrin; /** número de variables caracter del programa
                                principal **/
private int varRealesPrin; /** número de variables reales del programa
                              principal **/
private int varTextPrin; /** número de variables tipo text del programa
                            principal **/
private int lineasCodigo; /** número total de líneas de código **/
private int numIf; /** número de 'if' que aparecen en el programa
                      principal **/
private int numCase; /** número de 'case' que aparecen en el programa
                        principal **/
private int numRepeat; /** número de 'repeat' que aparecen en el programa
                          principal **/
private int numWhile; /** número de 'while' que aparecen en el programa
                         principal **/
private int numFor; /** número de 'for' que aparacen en el programa
                       principal **/
private int numAsig; /** número de asignaciones que aparecen en el programa
                        principal **/
private Vector arrayFunProc; /** vector de funciones/procedimientos
                                declarados en el programa principal
                                Los elementos de este vector son de tipo
                                clase FunProc **/

/** variables correspondientes al conjunto de lexemas del fichero a evaluar,
    y al token en curso **/
private int tokenActual;
private StreamTokenizer conjuntoLexemas;
```

Aunque no hubieran hecho falta tantos atributos pues se podían haber llevado de forma genérica por ejemplo el número de variables independientemente del tipo (pues para el nivel de aproximación no se tiene en cuenta), se decidió guardar la información de forma detallada, para generar ficheros de textos, que contuvieran dicha información, y así el usuario tuviera más nivel de detalle, tanto de su propia práctica, como de la del profesor. Estos ficheros de texto se explicaron con más detalle en el punto 4.2.3.-

Lo primero que se hace en esta clase, es convertir la practica que se va a tratar, en un conjunto de lexemas, es decir el fichero se convierte en un objeto de tipo **StreamTokenizer()**, lo que permitirá analizar cada una de las palabras de la práctica, y poder distinguir si son palabras reservadas o no. De esta forma se va tratando cada una de las partes de las que puede constituir un programa en Pascal, y se van actualizando los atributos de la clase **`Comparacion1()`** cuando sea necesario.

Destacar que para guardar la información de las funciones y procedimientos declarados se utiliza un array cuyos elementos son del tipo de la clase llamada **`FunProc()`**, que tiene la siguiente estructura:

```
public class FunProc {

  /** Esta clase guarda la información necesaria referente a subprogramas **/

  /** atributos de la clase **/
  private String nombre; /* nombre del subprograma */
  private boolean esFun; /* dice si el subprograma es una función o un
                            procedimiento. Si este atributo está a TRUE es que
                            el subprograma es una función; si esta a FALSE
                            entonces el subprograma es un procedimiento */
  private int numVar; /* número de variables declaradas en el subprograma */
  private Vector arrayFunProc; /* información acerca de las funciones y
                                  procedimientos declarados en el subprograma.
                                  Es decir, este vector guarda los subprogramas
                                  anidados, asi pues el tipo de los elementos
                                  será tambien de tipo FunProc */


  /** constructor de la clase **/
  public FunProc(boolean esF, int nVar) {
    esFun = esF;
    numVar = nVar;
    arrayFunProc = new Vector();
  }

  /* Métodos para devolver y actualizar los atributos necesarios de la clase */
  public void setNombre(String str) { nombre=str; }
  public void aniadeVariable (int num) { this.numVar = this.numVar+num; }

  public String getNombre () { return nombre; }
  public boolean getEsFun () { return esFun; }
  public int getNumVar () { return numVar; }
  public Vector getVectorFunProc () { return arrayFunProc; }

} /* fin de la clase FunProc */
```

Se utiliza para poder llevar la cuenta también de procedimientos y funciones anidados, (se puede ver que esta clase tiene un atributo, que es un array del tipo de la propia clase). De esta forma el array de ***FunProc*** que se encuentra declarado en la clase **`Comparacion1()`** lleva la información de los subprogramas declarados de forma paralela (a la misma altura) en el programa principal, y cada uno de esos subprogramas llevaría información acerca de los subprogramas que ellos mismos tuvieran anidados (cuando se diera el caso), y así sucesivamente.

Los métodos declarados en la clase **`Comparacion1()`** permiten detectar todas las partes que puede contener un programa escrito en Pascal, realizando las operaciones oportunas en cada caso (por ejemplo si se encuentran variables se tendrán que contar, pero si se encuentra una sección de declaración de constantes, podremos saltarla entera puesto que no es significativa a la hora de calcular el nivel de aproximación). Así, pues se tienen métodos tales como:

```
public int cuentaFun(Vector array)

/* Este método cuenta el número de funciones declaradas, o bien en el
   programa principal, o bien declaradas de forma anidada en alguna función
   o procedimiento. Depende del vector de funciones/procedimientos que se le
   pase como parámetro, dicho vector puede ser el vector de
   funciones/procedimientos del programa principal, o el vector de
   funciones/procedimientos de algún subprograma */
```

**public int cuentaProc(Vector array)**

```
/* Este método cuenta el número de procedimientos declarados, o bien en el
   programa principal, o bien declarados de forma anidada en alguna función
   o procedimiento. Depende del vector de funciones/procedimientos que se le
   pase como parámetro, dicho vector puede ser el vector de
   funciones/procedimientos del programa principal, o el vector de
   funciones/procedimientos de algún subprograma */
```

---

**public void cuentaVariables (FunProc funProc)**

```
/* Este método cuenta las variables declaradas en el programa principal (si
   el parámetro es null), o las variables declaradas en algún subprograma
   (si el parámetro de entrada es distinto de null).
   La declaracion de variables de forma genérica en Pascal es de la
   siguiente
   forma:
                    var
                      variab1,variab2, ..., variabN: tipo1;
                      variab11: tipo2;

   (teniendo en cuenta que puede haber finales de línea, o no, entre
   cualquiera de los tokens) */
```

---

**public void cuentaOperLogicos ()**

```
/* Este método actualiza el atributo operLogicos (que lleva la cuenta del
   número de operadores lógicos que hay en el programa principal) si el
   token actual se corresponde con un operador lógico (de tipo símbolo).
   Este tipo de operadores lógicos pueden ser: >, <, >=, <=, =, <> */
```

---

**public void cuentaInstrucciones() {**

```
 /* las posibles instrucciones son 'if', 'case', 'repeat', 'while','for' */
```

---

**public void cuentaFunProc (FunProc funcion)**

```
/** Este método testea todo el contenido de una función o un procedimiento
    que se le pasa como parámetro. La idea es que no sale del método hasta
    que no encuentra el 'end' correspondiente al cuerpo del subprograma.
    De esta forma testea tanto la parte de declaraciones (contando número
    general de variables y funciones/procedimientos declarados dentro del
    subprograma), y cuando llega al cuerpo del subprograma, no haría nada
    más que contar los finales de línea (las instrucciones no las cuenta
    porque solo estamos interesados en las instrucciones del programa
    principal **/
```

---

**public void saltaConstante()**

```
/* Este método se salta la sección correspondiente a la declaración de
   constantes */
```

---

```
public void declaraciones (FunProc funProc)

/* Este método comprueba la parte de declaraciones del programa principal
   (si el parametro de entrada es null), o bien cuenta la parte de las
   declaraciones de una funcion/procedimiento (si el parametro de entrada es
   distinto de null).
   No sale del método (es recursivo) hasta que no encuentra
   el 'begin' correspondiente al cuerpo (que contiene las instrucciones),
   de esta forma testea tanto la declaración de variables como las posibles
   funciones/procedimientos que hubieran declaradas.
   La idea es que en las declaraciones pueden aparacer en cualquier orden
   la declaracion de las variables y de funciones y procedimientos es decir
   se podría tener algo como:

        VAR declaracion_variables
        FUNCTION funcion
        VAR declaracion_variables2
        PROCEDURE proced
        FUNCTION funcion2

   o bien tener algo como:
        FUNCTION funcion1
        FUNCTION funcion2
        PROCEDURE proced
        VAR declaracion_variables
        .
        .
        .
    asi como cualquier tipo de combinación que se quiera con las
    declaraciones. */
```

### *5.2.3.1.c.- Comparación 2*

En una primera aproximación al problema se intentó convertir el fichero normalizado en un vector de instrucciones como sigue. Se leía del programa principal las instrucciones que estuvieran al mismo nivel, es decir aquellas que no estuvieran en el cuerpo de otras. Y de esas instrucciones se analizaban los términos de la instrucción, es decir si era un if, se analizaban los términos de la o las comparaciones, y a continuación todo el cuerpo del if con todas las instrucciones que llevaran anidadas se guardaban en un String. La idea era posteriormente al comparar las instrucciones de dos programas, ir comparando el cuerpo de cada una de ellas, para ver qué tenían y obtener el grado de similitud. Se implementó con esa idea, pero posteriormente a la hora de analizar el cuerpo de las instrucciones se plantearon muchas dificultades para al análisis y la comparación de los cuerpos. Se crearon las clases de las instrucciones con esa idea de manera que también si se quisiera se pudiera llevar a cabo un análisis de las condiciones de los if, las asignaciones, las condiciones de repetición de los bucles... pero dada la falta de tiempo y la complejidad del proceso, se descartó esta opción, si bien se han dejado creadas estructuras que pueden facilitar la continuación de este análisis en una futura ampliación del proyecto.

A continuación se decidió que la mejor manera de analizar las instrucciones era cambiando el cuerpo de las instrucciones de string a vectores de instrucciones y así de forma recursiva ir leyendo instrucción tras instrucción e introduciéndolas en el cuerpo correspondiente. Se utilizan las siguientes clases. La interfaz **Instrucción** que tiene como atributo el tipo de instrucción que es (IF, CASE, WHILE, REPEAT, ASIGNACIÓN, FOR o LLAMADA A FUNCIÓN). Todas las clases que servirán para guardar las instrucciones implementarán dicha interfaz. La clases son **Cwhile, Cif, Crepeat, Ccase, Cfor, Casignacion, CFunProc y CcaseCasos** (que implementará cada caso del case). A continuación pasamos a describir los atributos de cada clase.

**Cwhile:**

WHILE (TERM1 COMP TERM2) DO CUERPO

private String tipo; // tipo de la instrucción, será "While".
private String term1; // primer término de la comparación.
private String term2; // segundo término de la comparación.
private String comp; // comparación (<, >=, >, ....).
private Vector cuerpo; // cuerpo del while (todas las instrucciones que tiene).
private int numero; // Indica el número de instrucción dentro del orden del programa.

El atributo número está en todas las clases, pero pertenece a la estructura antigua. En la idea original se pretendía guardar vectores de instrucciones (un vector por cada tipo de instrucciones, uno para los ifs, unos para los while...) y asociar a cada instrucción su correspondiente ordinal, de manera que a la hora de ir sacando las instrucciones para la comparación, saber el orden de éstas. Se ha dejado el atributo por si se continúa en esta línea de trabajo en algún futuro.

**Ccase:**

CASE EXPRESION OF CASOS

private String tipo; // tipo de la instrucción, será "Case".
private String expresion; // expresión a evaluar.
private Vector casos; // vector con todos los casos de tipo CcaseCasos.
private int numero;

**CcaseCasos:**

CONDICIÓN: CUERPO

private String condicion; // valor de la expresión.
private Vector cuerpo; // acciones a llevar a cabo.

**Cfor:**

FOR (VALORINI TO VALOR FIN) DO CUERPO

```
private String tipo; // tipo de la instrucción, será "For".
private String valorIni; // valor inicial del conteo.
private String to; // puede ser "to" o "downto", sentido de la cuenta.
private String valorFin; // valor final del conteo.
private Vector cuerpo; // cuerpo del for.
private int numero;
```

**Crepeat:**

REPEAT (CUERPO) UNTIL CONDICION

```
private String tipo;  // Tipo de la instrucción, será “Repeat”.
private String condicion;  // condición de repetición.
private Vector cuerpo;  // cuerpo del repeat.
private int numero;
```

**Casignacion:**

TERM1 := TERM2;

```
private String tipo;  // tipo de la instrucción, en este caso será “Asignación”.
private String term1;  // variable a la que se asigna un valor.
private String term2; // valor o expresión que se asigna a la variable.
private int numero;
```

**CFunProc:**

FUNCTION NOMBRE(PARAMETROS);
PROCEDURE NOMBRE(PARAMETROS);

```
private String tipo;  // Tipo de la instrucción, es est caso será  “Función”.
private boolean esFun;  // true si es un afunción o false si es un procedimiento.
private String nombre;  // nombre de la función o precedimiento.
private Vector parametros; // lista de parámetros de la función o procedimiento.
private int numero;
```

**Cif:**

IF (TERM1 COMP TERM2) THEN SI ELSE OTRO

```
private String tipo; // tipo de la instrucción, en este caso será “If”.
private String term1; // término 1 de la comparación. (en caso de que solo haya uno por
                      // ser un booleano, el segundo estará en blanco).
private String term2  // término 2 de la comparación.;
private String comp; // tipo de comparación (<, >=, ...).
private Vector si;  // cuerpo del then, en caso sea cierta la comdición.
private Vector otro;  // cuerpo del else, en caso no se cumpla la condición.
private int numero;
```

- PROCESO DE COMPARACIÓN 2

En primer lugar se pasa a analizar sintácticamente el código ya normalizado. Se parte del fichero normalizado y se va leyendo el archivo de manera que se va creando un vector de instrucciones. Comenzamos analizando token a token el fichero, hasta llegar al programa principal. Una vez ahí, se sigue analizando hasta encontrar alguna palabra "clave" (es decir, el comienzo de una instrucción de las tratadas IF, WHILE, REPEAT, CASE, FOR, ASIGNACIÓN o LLAMADA A PROCEDIMIETNO O FUNCIÓN, ignorando los mensajes por pantalla, las aperturas o cierres de fichero y todo tipo de instrucciones alternativas). Con la palabra clave encontrada se pasa al analizador de instrucciones en donde se trata el cuerpo de cada instrucción. Para ello se vuelve a llamar al analizador de las instrucciones con la primera palabra clave encontrada, y así sucesivamente, hasta terminar con todas las instrucciones.

Una vez analizados los dos programas, hemos obtenido un vector de instrucciones para cada uno. Las instrucciones se encuentran en el vector en el orden en que aparecen en cada programa. Lo que comprobamos en esta segunda comparación es que las estructuras de ambos programas sean equivalentes.

Hemos supuesto que un programa debe comenzar con una serie de inicializaciones de variables, igualmente comenzarán los cuerpos de las instrucciones de control con inicializaciones. Las instrucciones de control deberán seguir un orden, que será el del profesor, ya que esa será la lógica del programa.

Vamos comparando instrucción a instrucción de los vectores obtenidos en el análisis de los archivos. Comenzamos comprobando las inicializaciones, tal como hemos dicho antes. Al encontrar una instrucción de control (IF, WHILE,..) comparamos sus cuerpos recursivamente.

Veamos como hemos baremado esta comparación:

- Para las inicializaciones: se cuentan el número de inicializaciones (asignaciones a variables) de cada vector. Si son iguales, entonces la inicialización es equivalente, si hay diferente número de inicializaciones, se penaliza con el valor absoluto de la diferencia, asumiendo que esta diferencia es debida a que el alumno no ha declarado las suficientes variables, o ha declarado de más.
- Para la instrucción de control IF: Si encontramos que el profesor tiene una instrucción de control IF, el alumno debe de tener también un IF, ya que no hemos supuesto ninguna instrucción equivalente a esta. En el caso de que el alumno tenga también un IF no se le penalizará, en caso contrario se le penalizará con cinco puntos, ya que su instrucción no es equivalente.
- Para la instrucción WHILE: Para esta instrucción hemos supuesto que son equivalentes la instrucción FOR y la instrucción REPEAT. Si el alumno tiene, al igual que el profesor una instrucción WHILE no se le penalizará, si por el contrario tiene un FOR o un REPEAT se le penalizará con un punto, ya que las instrucciones son equivalentes, pero no son la misma instrucción de control encontrada para el profesor. En el caso de que el alumno no tenga ninguna de las instrucciones equivalentes, se le penalizará con cinco puntos.
- Para la instrucción FOR: Para esta instrucción hemos supuesto que son equivalentes a ella la instrucción WHILE y la instrucción REPEAT. Al igual que en el caso del WHILE, si el alumno tiene una de las dos instrucciones equivalentes, se le penalizará con un punto; y en caso de que no sea ninguna de las tres (FOR o las dos equivalentes) se le penalizará con cinco puntos.

- Para la instrucción REPEAT: Para esta instrucción hemos supuesto que son equivalentes a ella la instrucción WHILE y la instrucción FOR. Al igual que en los dos casos anteriores no se le penalizará si tiene un REPEAT, se le penalizará con un punto si tiene cualquiera de las dos instrucciones equivalentes, y con cinco puntos si la instrucción no es equivalente.
- Para la instrucción CASE: No se ha supuesto que sea equivalente a ninguna otra instrucción de control. Si el alumno tiene un CASE no se le penalizará, en caso contrario se le penalizará con 5 puntos.

Cuando se han vaciado el vector de instrucciones del profesor acabamos la comparación. Si quedan todavía instrucciones del alumno por tratar se le penaliza con el número de instrucciones sobrantes.

Todas las clases relacionadas con esta comparación pertenecen a un paquete llamado comparacion2. La clase Comparacion2 se encarga de analizar los archivos y crear el vector de instrucciones asociado a cada uno. La clase Comparaciones es la encargada de comparar los dos vectores de instrucciones, el del profesor y el del alumno. Tiene un atributo privado nivel, en el que se guarda el nivel obtenido.

Una vez realizadas las dos comparaciones, se suman los dos niveles obtenidos en cada una, y ese será el nivel de aproximación del alumno.

# 6.- EJEMPLOS DE APLICACIÓN

El sistema se ha probado sobre una serie de prácticas realizadas por algunos alumnos de primero de la licenciatura de CC. Matemáticas. El código de las prácticas se encuentra en el apéndice I.

## 6.1.-PRUEBAS DE PRÁCTICA DE GENERACIÓN DEL CÓDIGO ASCII

Para estas pruebas disponemos de doce prácticas entregadas por los alumnos con su correspondiente ejecutable y de la práctica del profesor para realizar las comparaciones pertinentes.

Esta práctica no necesita archivos de entrada para ejecutarse, ya que se limita a escribir en un archivo la tabla de caracteres ASCII. En nuestro programa no contemplamos la opción de que los programas no tuvieran entrada, por lo que siempre hay que elegir entre generación automática o manual de datos. En este caso la ejecución será la misma para los dos tipos de generaciones de datos, ya que la práctica no usa datos de entrada.

### 6.1.1.- Ejecución de los *.exe* de los alumnos

Como se dijo anteriormente, las ejecuciones con datos generados automáticamente y datos generados manualmente serán equivalentes, por lo que en este caso se tomarán los resultados de la ejecución con datos generados automáticamente.

Se realizó una ejecución, ya que la salida de la práctica no depende de los datos de entrada, que además en este caso no tiene.

```
-- RESULTADO DE LA EJECUCIÓN --


--------------------------------------------------------------------------------
- RESULTADOS PARA LOS DATOS DE ENTRADA QUE ESTÁN EN EL DIRECTORIO:
C:\arma\Ascii\GA_Entradas\ Y QUE EMPIEZAN POR GA1

EJECUTA: P3D1347678.EXE
LA SALIDA QUE GENERA LA EJECUCIÓN SE PUEDE ENCONTRAR EN
C:\arma\Ascii\GA_Salida1\P3D1347678\

EJECUTA: P3D1354647.EXE
LA SALIDA QUE GENERA LA EJECUCIÓN SE PUEDE ENCONTRAR EN
C:\arma\Ascii\GA_Salida1\P3D1354647\

EJECUTA: P3D1425678.EXE
LA SALIDA QUE GENERA LA EJECUCIÓN SE PUEDE ENCONTRAR EN
C:\arma\Ascii\GA_Salida1\P3D1425678\

EJECUTA: P3D1456787.EXE
LA SALIDA QUE GENERA LA EJECUCIÓN SE PUEDE ENCONTRAR EN
C:\arma\Ascii\GA_Salida1\P3D1456787\

EJECUTA: P3D213564.EXE
LA SALIDA QUE GENERA LA EJECUCIÓN SE PUEDE ENCONTRAR EN
C:\arma\Ascii\GA_Salida1\P3D213564\
```

EJECUTA: P3D23432627.EXE
LA SALIDA QUE GENERA LA EJECUCIÓN SE PUEDE ENCONTRAR EN C:\arma\Ascii\GA_Salida1\P3D23432627\

EJECUTA: P3D324532.EXE
LA SALIDA QUE GENERA LA EJECUCIÓN SE PUEDE ENCONTRAR EN C:\arma\Ascii\GA_Salida1\P3D324532\

EJECUTA: P3D34354567.EXE
LA SALIDA QUE GENERA LA EJECUCIÓN SE PUEDE ENCONTRAR EN C:\arma\Ascii\GA_Salida1\P3D34354567\

EJECUTA: P3D346778.EXE
LA SALIDA QUE GENERA LA EJECUCIÓN SE PUEDE ENCONTRAR EN C:\arma\Ascii\GA_Salida1\P3D346778\

EJECUTA: P3D4564567.EXE
LA SALIDA QUE GENERA LA EJECUCIÓN SE PUEDE ENCONTRAR EN C:\arma\Ascii\GA_Salida1\P3D4564567\

EJECUTA: P3D54546765.EXE
LA SALIDA QUE GENERA LA EJECUCIÓN SE PUEDE ENCONTRAR EN C:\arma\Ascii\GA_Salida1\P3D54546765\

EJECUTA: P3D56347845.EXE
LA SALIDA QUE GENERA LA EJECUCIÓN SE PUEDE ENCONTRAR EN C:\arma\Ascii\GA_Salida1\P3D56347845\

Como se puede observar en el archivo generado después de la ejecución de las prácticas, ejecutan todas ellas, es decir, el 100% de las prácticas disponibles de los alumnos.

Las salidas que genera cada práctica que sí ejecuta son las siguientes:

*P3D1347678.exe*

Genera un fichero de salida vacío.

*P3D1354647.exe*

□□□□□□□□□□□□□□□□
□□□□□□□□□□- !"#$%&'(
)*+,-./0123456789:;<
=>?@ABCDEFGHIJKLMNOP
QRSTUVWXYZ[\]^_`abcd
efghijklmnopqrstuvwx
yz{|}~□€□‚ƒ„…†‡ˆ‰Š‹Œ
□Ž□□‘’“”•–—˜™š›œ□žŸ
¡¢£¤¥¦§¨©ª«¬-®¯°±²³´
µ¶·¸¹º»¼½¾¿ÀÁÂÃÄÅÆÇÈ
ÉÊËÌÍÎÏÐÑÒÓÔÕÖ×ØÙÚÛÜ
ÝÞßàáâãäåæçèéêëìíîï
ñòóôõö÷øùúûüýþÿ

*P3D1425678.exe*

```
□ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □
□ □ □ □ □ □ □ □ □ □ -   ! " # $ % & ' (
) * + , - . / 0 1 2 3 4 5 6 7 8 9 : ; <
= > ? @ A B C D E F G H I J K L M N O P
Q R S T U V W X Y Z [ \ ] ^ _ ` a b c d
e f g h i j k l m n o p q r s t u v w x
y z { | } ~ □ € □ , ƒ „ … † ‡ ˆ ‰ Š ‹ Œ
□ Ž □ □ ‘ ’ “ ” • – — ˜ ™ š › œ □ ž Ÿ
¡ ¢ £ ¤ ¥ ¦ § ¨ © ª « ¬ - ® ¯ ° ± ² ³ ´
µ ¶ · ¸ ¹ º » ¼ ½ ¾ ¿ À Á Â Ã Ä Å Æ Ç È
É Ê Ë Ì Í Î Ï Ð Ñ Ò Ó Ô Õ Ö × Ø Ù Ú Û Ü
Ý Þ ß à á â ã ä å æ ç è é ê ë ì í î ï ð
ñ ò ó ô õ ö ÷ ø
```

*P3D1456787.exe*

```
fila 1 □ □ □ □ □ □ □ □ □ □ □ □ □ □ □
fila 2 □ □ □ □ □ □ □ □ □ □ -   ! " # $ % & ' (
fila 3 ) * + , - . / 0 1 2 3 4 5 6 7 8 9 : ; <
fila 4 = > ? @ A B C D E F G H I J K L M N O P
fila 5 Q R S T U V W X Y Z [ \ ] ^ _ ` a b c d
fila 6 e f g h i j k l m n o p q r s t u v w x
fila 7 y z { | } ~ □ € □ , ƒ „ … † ‡ ˆ ‰ Š ‹ Œ
fila 8 □ Ž □ □ ‘ ’ “ ” • – — ˜ ™ š › œ □ ž Ÿ
fila 9 ¡ ¢ £ ¤ ¥ ¦ § ¨ © ª « ¬ - ® ¯ ° ± ² ³ ´
fila 10 µ ¶ · ¸ ¹ º » ¼ ½ ¾ ¿ À Á Â Ã Ä Å Æ Ç È
fila 11 É Ê Ë Ì Í Î Ï Ð Ñ Ò Ó Ô Õ Ö × Ø Ù Ú Û Ü
fila 12 Ý Þ ß à á â ã ä å æ ç è é ê ë ì í î ï ð
fila 13 ñ ò ó ô õ ö ÷ ø ù ú û ü ý þ ÿ
```

*P3D213564.exe*

```
Fila 1 □ □ □ □ □ □ □ □ □ □ □ □ □ □ □
Fila 2 □ □ □ □ □ □ □ □ □ □ -   ! " # $ % & ' '
Fila 3 ) * + , - . / 0 1 2 3 4 5 6 7 8 9 : ; ;
Fila 4 = > ? @ A B C D E F G H I J K L M N O O
Fila 5 Q R S T U V W X Y Z [ \ ] ^ _ ` a b c c
Fila 6 e f g h i j k l m n o p q r s t u v w w
Fila 7 y z { | } ~ □ € □ , ƒ „ … † ‡ ˆ ‰ Š ‹ ‹
Fila 8 □ Ž □ □ ‘ ’ “ ” • – — ˜ ™ š › œ □ ž Ÿ Ÿ
Fila 9 ¡ ¢ £ ¤ ¥ ¦ § ¨ © ª « ¬ - ® ¯ ° ± ² ³ ³
Fila 10 µ ¶ · ¸ ¹ º » ¼ ½ ¾ ¿ À Á Â Ã Ä Å Æ Ç Ç
Fila 11 É Ê Ë Ì Í Î Ï Ð Ñ Ò Ó Ô Õ Ö × Ø Ù Ú Û Û
Fila 12 Ý Þ ß à á â ã ä å æ ç è é ê ë ì í î ï ï
Fila 13 ñ ò ó ô õ ö ÷ ø ù ú û ü ý þ ÿ
```

*P3D23432627.exe*

```
fila1  □ □ □ □ □ □ □ □ □ □ □ □ □ □
fila2 □ □ □ □ □ □ □ □ □ □ □ -  ! " # $ % & '
fila3 ( ) * + , - . / 0 1 2 3 4 5 6 7 8 9 : ;
fila4 < = > ? @ A B C D E F G H I J K L M N O
fila5 P Q R S T U V W X Y Z [ \ ] ^ _ ` a b c
fila6 d e f g h i j k l m n o p q r s t u v w
fila7 x y z { | } ~ □ € □ , ƒ „ … † ‡ ˆ ‰ Š ‹
fila8 Œ □ Ž □ □ ‘ ’ “ ” • – — ˜ ™ š › œ □ ž Ÿ
fila9  ¡ ¢ £ ¤ ¥ ¦ § ¨ © ª « ¬ - ® ¯ ° ± ² ³
fila10 ´ µ ¶ · ¸ ¹ º » ¼ ½ ¾ ¿ À Á Â Ã Ä Å Æ Ç
fila11 È É Ê Ë Ì Í Î Ï Ð Ñ Ò Ó Ô Õ Ö × Ø Ù Ú Û
fila12 Ü Ý Þ ß à á â ã ä å æ ç è é ê ë ì í î ï
fila13 ð ñ ò ó ô õ ö ÷ ø ù ú û ü ý þ ÿ
```

*P3D324532.exe*

```
fila0  □ □ □ □ □ □ □ □ □ □ □ □ □ □
fila1 □ □ □ □ □ □ □ □ □ □ □ -  ! " # $ % & '
fila2 ( ) * + , - . / 0 1 2 3 4 5 6 7 8 9 : ;
fila3 < = > ? @ A B C D E F G H I J K L M N O
fila4 P Q R S T U V W X Y Z [ \ ] ^ _ ` a b c
fila5 d e f g h i j k l m n o p q r s t u v w
fila6 x y z { | } ~ □ € □ , ƒ „ … † ‡ ˆ ‰ Š ‹
fila7 Œ □ Ž □ □ ‘ ’ “ ” • – — ˜ ™ š › œ □ ž Ÿ
fila8  ¡ ¢ £ ¤ ¥ ¦ § ¨ © ª « ¬ - ® ¯ ° ± ² ³
fila9 ´ µ ¶ · ¸ ¹ º » ¼ ½ ¾ ¿ À Á Â Ã Ä Å Æ Ç
fila10 È É Ê Ë Ì Í Î Ï Ð Ñ Ò Ó Ô Õ Ö × Ø Ù Ú Û
fila11 Ü Ý Þ ß à á
```

*P3D34354567.exe*

```
Fila 1  □ □ □ □ □ □ □ □ □ □ □ □ □ □
Fila 2  □□ □ □ □ □ □ □ □ □ -  ! " # $ % & ' (
Fila 3  () * + , - . / 0 1 2 3 4 5 6 7 8 9 : ; <
Fila 4  <= > ? @ A B C D E F G H I J K L M N O P
Fila 5  PQ R S T U V W X Y Z [ \ ] ^ _ ` a b c d
Fila 6  de f g h i j k l m n o p q r s t u v w x
Fila 7  xy z { | } ~ □ € □ , ƒ „ … † ‡ ˆ ‰ Š ‹ Œ
Fila 8  Œ□ Ž □ □ ‘ ’ “ ” • – — ˜ ™ š › œ □ ž Ÿ
Fila 9  ¡ ¢ £ ¤ ¥ ¦ § ¨ © ª « ¬ - ® ¯ ° ± ² ³ ´
Fila 10  ´µ ¶ · ¸ ¹ º » ¼ ½ ¾ ¿ À Á Â Ã Ä Å Æ Ç È
Fila 11  ÈÉ Ê Ë Ì Í Î Ï Ð Ñ Ò Ó Ô Õ Ö × Ø Ù Ú Û Ü
Fila 12  ÜÝ Þ ß à á â ã ä å æ ç è é ê ë ì í î ï ð
Fila 13  ðñ ò ó ô õ ö ÷ ø ù ú û ü ý þ ÿ
```

*P3D346778.exe*

```
Fila 1  □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □
fila 2 □ □ □ □ □ □ □ □ □ □ -  ! " # $ % & ' ( )
fila 3 * + , - . / 0 1 2 3 4 5 6 7 8 9 : ; < = >
fila 4 ? @ A B C D E F G H I J K L M N O P Q R S
fila 5 T U V W X Y Z [ \ ] ^ _ ` a b c d e f g h
fila 6 i j k l m n o p q r s t u v w x y z { | }
fila 7 ~ □ € □ , ƒ „ … † ‡ ˆ ‰ Š ‹ Œ □ Ž □ □ ‘ ’
fila 8 “ ” • – — ˜ ™ š › œ □ ž Ÿ  ¡ ¢ £ ¤ ¥ ¦ §
fila 9 ¨ © ª « ¬ - ® ¯ ° ± ² ³ ´ µ ¶ · ¸ ¹ º » ¼
fila 10 ½ ¾ ¿ À Á Â Ã Ä Å Æ Ç È É Ê Ë Ì Í Î Ï Ð Ñ
fila 11 Ò Ó Ô Õ Ö × Ø Ù Ú Û Ü Ý Þ ß à á â ã ä å æ
fila 12 ç è é ê ë ì í î ï ð ñ ò ó ô õ ö ÷ ø ù ú û
fila 13 ü ý þ ÿ
```

*P3D4564567.exe*

```
fila1 □ □ □ □ □ □ □ □ □ □ □ □ □ □
fila2 □ □ □ □ □ □ □ □ □ □ □ -  ! " # $ % & '
fila3 ( ) * + , - . / 0 1 2 3 4 5 6 7 8 9 : ;
fila4 < = > ? @ A B C D E F G H I J K L M N O
fila5 P Q R S T U V W X Y Z [ \ ] ^ _ ` a b c
fila6 d e f g h i j k l m n o p q r s t u v w
fila7 x y z { | } ~ □ € □ , ƒ „ … † ‡ ˆ ‰ Š ‹
fila8 Œ □ Ž □ □ ‘ ’ “ ” • – — ˜ ™ š › œ □ ž Ÿ
fila9  ¡ ¢ £ ¤ ¥ ¦ § ¨ © ª « ¬ - ® ¯ ° ± ² ³
fila10 ´ µ ¶ · ¸ ¹ º » ¼ ½ ¾ ¿ À Á Â Ã Ä Å Æ Ç
fila11 È É Ê Ë Ì Í Î Ï Ð Ñ Ò Ó Ô Õ Ö × Ø Ù Ú Û
fila12 Ü Ý Þ ß à á â ã ä å æ ç è é ê ë ì í î ï
fila13 ð ñ ò ó ô õ ö ÷ ø ù ú û ü ý þ ÿ
```

*P3D54546765.exe*

```
fila1  □ □ □ □ □ □ □ □ □ □ □ □ □ □
fila2 □ □ □ □ □ □ □ □ □ □ □ -  ! " # $ % & '
fila3 ( ) * + , - . / 0 1 2 3 4 5 6 7 8 9 : ;
fila4 < = > ? @ A B C D E F G H I J K L M N O
fila5 P Q R S T U V W X Y Z [ \ ] ^ _ ` a b c
fila6 d e f g h i j k l m n o p q r s t u v w
fila7 x y z { | } ~ □ € □ , ƒ „ … † ‡ ˆ ‰ Š ‹
fila8 Œ □ Ž □ □ ‘ ’ “ ” • – — ˜ ™ š › œ □ ž Ÿ
fila9  ¡ ¢ £ ¤ ¥ ¦ § ¨ © ª « ¬ - ® ¯ ° ± ² ³
fila10 ´ µ ¶ · ¸ ¹ º » ¼ ½ ¾ ¿ À Á Â Ã Ä Å Æ Ç
fila11 È É Ê Ë Ì Í Î Ï Ð Ñ Ò Ó Ô Õ Ö × Ø Ù Ú Û
fila12 Ü Ý Þ ß à á â ã ä å æ ç è é ê ë ì í î ï
fila13 ð ñ ò ó ô õ ö ÷ ø ù ú û ü ý þ ÿ
```

*P3D56347845.exe*

```
Fila 1   □ □ □ □ □ □ □ □ □ □ □ □ □ □
Fila 2 □ □ □ □ □ □ □ □ □ □ □ -    ! " # $ % & '
Fila 3 ( ) * + , - . / 0 1 2 3 4 5 6 7 8 9 : ;
Fila 4 < = > ? @ A B C D E F G H I J K L M N O
Fila 5 P Q R S T U V W X Y Z [ \ ] ^ _ ` a b c
Fila 6 d e f g h i j k l m n o p q r s t u v w
Fila 7 x y z { | } ~ □ € □ , ƒ „ … † ‡ ˆ ‰ Š ‹
Fila 8 Œ □ Ž □ □ ‘ ’ “ ” • – — ˜ ™ š › œ □ ž Ÿ
Fila 9   ¡ ¢ £ ¤ ¥ ¦ § ¨ © ª « ¬ - ® ¯ ° ± ² ³
Fila 10 ´ µ ¶ · ¸ ¹ º » ¼ ½ ¾ ¿ À Á Â Ã Ä Å Æ Ç
Fila 11 È É Ê Ë Ì Í Î Ï Ð Ñ Ò Ó Ô Õ Ö × Ø Ù Ú Û
Fila 12 Ü Ý Þ ß à á â ã ä å æ ç è é ê ë ì í î ï
Fila 13 ð ñ ò ó ô õ ö ÷ ø ù ú û ü ý þ ÿ
```

*P3PROF.exe*

```
Fila 1.  □ □ □ □ □ □ □ □ □ □ □ □ □ □
Fila 2.  □ □ □ □ □ □ □ □ □ □ -    ! " # $ % & ' (
Fila 3.  ) * + , - . / 0 1 2 3 4 5 6 7 8 9 : ; <
Fila 4.  = > ? @ A B C D E F G H I J K L M N O P
Fila 5.  Q R S T U V W X Y Z [ \ ] ^ _ ` a b c d
Fila 6.  e f g h i j k l m n o p q r s t u v w x
Fila 7.  y z { | } ~ □ € □ , ƒ „ … † ‡ ˆ ‰ Š ‹ Œ
Fila 8.  □ Ž □ □ ‘ ’ “ ” • – — ˜ ™ š › œ □ ž Ÿ
Fila 9.  ¡ ¢ £ ¤ ¥ ¦ § ¨ © ª « ¬ - ® ¯ ° ± ² ³ ´
Fila 10.  µ ¶ · ¸ ¹ º » ¼ ½ ¾ ¿ À Á Â Ã Ä Å Æ Ç È
Fila 11.  É Ê Ë Ì Í Î Ï Ð Ñ Ò Ó Ô Õ Ö × Ø Ù Ú Û Ü
Fila 12.  Ý Þ ß à á â ã ä å æ ç è é ê ë ì í î ï ð
Fila 13.  ñ ò ó ô õ ö ÷ ø ù ú û ü ý þ ÿ
```

## 6.1.2.- Compilación de las prácticas

Se compilaron las prácticas y se obtuvieron los siguientes resultados:

```
-- RESULTADO DE LA COMPILACIÓN --

COMPILA: P3D1347678.pas

COMPILA: P3D1354647.pas

COMPILA: P3D1425678.pas

COMPILA: P3D1456787.pas

COMPILA: P3D213564.pas

COMPILA: P3D23432627.pas

COMPILA: P3D324532.pas
```

COMPILA: P3D34354567.pas

COMPILA: P3D346778.pas

COMPILA: P3D4564567.pas

COMPILA: P3D54546765.pas

COMPILA: P3D56347845.pas

Como se puede observar, compilan el 100% de las prácticas disponibles.

### 6.1.3.- Ejecución del *.exe* generado por el código fuente del alumno

Los resultados de la ejecución del *.exe* generado a partir del código fuente del alumno son los siguientes:

-- RESULTADO DE LA EJECUCIÓN --

---

- RESULTADOS PARA LOS DATOS DE ENTRADA QUE ESTÁN EN EL DIRECTORIO: C:\arma\Ascii\Compiladas\ExeGenerados\GA_Entradas\ Y QUE EMPIEZAN POR GA1

EJECUTA: P3D1347678.EXE
LA SALIDA QUE GENERA LA EJECUCIÓN SE PUEDE ENCONTRAR EN C:\arma\Ascii\Compiladas\ExeGenerados\GA_Salida1\P3D1347678\

EJECUTA: P3D1354647.EXE
LA SALIDA QUE GENERA LA EJECUCIÓN SE PUEDE ENCONTRAR EN C:\arma\Ascii\Compiladas\ExeGenerados\GA_Salida1\P3D1354647\

EJECUTA: P3D1425678.EXE
LA SALIDA QUE GENERA LA EJECUCIÓN SE PUEDE ENCONTRAR EN C:\arma\Ascii\Compiladas\ExeGenerados\GA_Salida1\P3D1425678\

EJECUTA: P3D1456787.EXE
LA SALIDA QUE GENERA LA EJECUCIÓN SE PUEDE ENCONTRAR EN C:\arma\Ascii\Compiladas\ExeGenerados\GA_Salida1\P3D1456787\

EJECUTA: P3D213564.EXE
LA SALIDA QUE GENERA LA EJECUCIÓN SE PUEDE ENCONTRAR EN C:\arma\Ascii\Compiladas\ExeGenerados\GA_Salida1\P3D213564\

EJECUTA: P3D23432627.EXE
LA SALIDA QUE GENERA LA EJECUCIÓN SE PUEDE ENCONTRAR EN C:\arma\Ascii\Compiladas\ExeGenerados\GA_Salida1\P3D23432627\

EJECUTA: P3D324532.EXE
LA SALIDA QUE GENERA LA EJECUCIÓN SE PUEDE ENCONTRAR EN C:\arma\Ascii\Compiladas\ExeGenerados\GA_Salida1\P3D324532\

EJECUTA: P3D34354567.EXE
LA SALIDA QUE GENERA LA EJECUCIÓN SE PUEDE ENCONTRAR EN C:\arma\Ascii\Compiladas\ExeGenerados\GA_Salida1\P3D34354567\

EJECUTA: P3D346778.EXE
LA SALIDA QUE GENERA LA EJECUCIÓN SE PUEDE ENCONTRAR EN C:\arma\Ascii\Compiladas\ExeGenerados\GA_Salida1\P3D346778\

EJECUTA: P3D4564567.EXE
LA SALIDA QUE GENERA LA EJECUCIÓN SE PUEDE ENCONTRAR EN C:\arma\Ascii\Compiladas\ExeGenerados\GA_Salida1\P3D4564567\

EJECUTA: P3D54546765.EXE
LA SALIDA QUE GENERA LA EJECUCIÓN SE PUEDE ENCONTRAR EN C:\arma\Ascii\Compiladas\ExeGenerados\GA_Salida1\P3D54546765\

EJECUTA: P3D56347845.EXE
LA SALIDA QUE GENERA LA EJECUCIÓN SE PUEDE ENCONTRAR EN C:\arma\Ascii\Compiladas\ExeGenerados\GA_Salida1\P3D56347845\

Los *.exe* generados por la compilación de las prácticas, ejecutan también todos, es decir, el 100% de ellos.

Las salidas que genera cada práctica que sí ejecuta son las siguientes:

*P3D1347678.exe*

Genera un fichero de salida vacío.

*P3D1354647.exe*

□□□□□□□□□□□□□□□□
□□□□□□□□□□- !"#$%&'(
)*+,-./0123456789:;<
=>?@ABCDEFGHIJKLMNOP
QRSTUVWXYZ[\]^_`abcd
efghijklmnopqrstuvwx
yz{|}~□€□‚ƒ„…†‡ˆ‰Š‹Œ
□Ž□□‘’“”•–—˜™š›œ□žŸ
¡¢£¤¥¦§¨©ª«¬-®¯°±²³´
µ¶·¸¹º»¼½¾¿ÀÁÂÃÄÅÆÇÈ
ÉÊËÌÍÎÏÐÑÒÓÔÕÖ×ØÙÚÛÜ
ÝÞßàáâãäåæçèéêëìíîïð
ñòóôõö÷øùúûüýþÿ

*P3D1425678.exe*

```
□ □ □ □ □ □ □ □ □ □ □ □ □ □ □
□ □ □ □ □ □ □ □ □ □ -  ! " # $ % & ' (
) * + , - . / 0 1 2 3 4 5 6 7 8 9 : ; <
= > ? @ A B C D E F G H I J K L M N O P
Q R S T U V W X Y Z [ \ ] ^ _ ` a b c d
e f g h i j k l m n o p q r s t u v w x
y z { | } ~ □ € □ , ƒ „ … † ‡ ˆ ‰ Š ‹ Œ
□ Ž □ □ ‘ ’ “ ” • – — ˜ ™ š › œ □ ž Ÿ
¡ ¢ £ ¤ ¥ ¦ § ¨ © ª « ¬ - ® ¯ ° ± ² ³ ´
µ ¶ · ¸ ¹ º » ¼ ½ ¾ ¿ À Á Â Ã Ä Å Æ Ç È
É Ê Ë Ì Í Î Ï Ð Ñ Ò Ó Ô Õ Ö × Ø Ù Ú Û Ü
Ý Þ ß à á â ã ä å æ ç è é ê ë ì í î ï ð
ñ ò ó ô õ ö ÷ ø
```

*P3D1456787.exe*

```
fila 1 □ □ □ □ □ □ □ □ □ □ □ □ □ □ □
fila 2 □ □ □ □ □ □ □ □ □ □ -  ! " # $ % & ' (
fila 3 ) * + , - . / 0 1 2 3 4 5 6 7 8 9 : ; <
fila 4 = > ? @ A B C D E F G H I J K L M N O P
fila 5 Q R S T U V W X Y Z [ \ ] ^ _ ` a b c d
fila 6 e f g h i j k l m n o p q r s t u v w x
fila 7 y z { | } ~ □ € □ , ƒ „ … † ‡ ˆ ‰ Š ‹ Œ
fila 8 □ Ž □ □ ‘ ’ “ ” • – — ˜ ™ š › œ □ ž Ÿ
fila 9 ¡ ¢ £ ¤ ¥ ¦ § ¨ © ª « ¬ - ® ¯ ° ± ² ³ ´
fila 10 µ ¶ · ¸ ¹ º » ¼ ½ ¾ ¿ À Á Â Ã Ä Å Æ Ç È
fila 11 É Ê Ë Ì Í Î Ï Ð Ñ Ò Ó Ô Õ Ö × Ø Ù Ú Û Ü
fila 12 Ý Þ ß à á â ã ä å æ ç è é ê ë ì í î ï ð
fila 13 ñ ò ó ô õ ö ÷ ø ù ú û ü ý þ ÿ
```

*P3D213564.exe*

```
Fila 1 □ □ □ □ □ □ □ □ □ □ □ □ □ □ □
Fila 2 □ □ □ □ □ □ □ □ □ □ -  ! " # $ % & ' '
Fila 3 ) * + , - . / 0 1 2 3 4 5 6 7 8 9 : ; ;
Fila 4 = > ? @ A B C D E F G H I J K L M N O O
Fila 5 Q R S T U V W X Y Z [ \ ] ^ _ ` a b c c
Fila 6 e f g h i j k l m n o p q r s t u v w w
Fila 7 y z { | } ~ □ € □ , ƒ „ … † ‡ ˆ ‰ Š ‹ ‹
Fila 8 □ Ž □ □ ‘ ’ “ ” • – — ˜ ™ š › œ □ ž Ÿ Ÿ
Fila 9 ¡ ¢ £ ¤ ¥ ¦ § ¨ © ª « ¬ - ® ¯ ° ± ² ³ ³
Fila 10 µ ¶ · ¸ ¹ º » ¼ ½ ¾ ¿ À Á Â Ã Ä Å Æ Ç Ç
Fila 11 É Ê Ë Ì Í Î Ï Ð Ñ Ò Ó Ô Õ Ö × Ø Ù Ú Û Û
Fila 12 Ý Þ ß à á â ã ä å æ ç è é ê ë ì í î ï ï
Fila 13 ñ ò ó ô õ ö ÷ ø ù ú û ü ý þ ÿ
```

*P3D23432627.exe*

fila1 □ □ □ □ □ □ □ □ □ □ □ □ □ □
fila2 □ □ □ □ □ □ □ □ □ □ □ - ! " # $ % & '
fila3 ( ) * + , - . / 0 1 2 3 4 5 6 7 8 9 : ;
fila4 < = > ? @ A B C D E F G H I J K L M N O
fila5 P Q R S T U V W X Y Z [ \ ] ^ _ ` a b c
fila6 d e f g h i j k l m n o p q r s t u v w
fila7 x y z { | } ~ □ € □ , *f* „ … † ‡ ˆ ‰ Š ‹
fila8 Œ □ Ž □ □ ‘ ’ “ ” • – — ˜ ™ š › œ □ ž Ÿ
fila9 ¡ ¢ £ ¤ ¥ ¦ § ¨ © ª « ¬ - ® ¯ ° ± ² ³
fila10 ´ µ ¶ · ¸ ¹ º » ¼ ½ ¾ ¿ À Á Â Ã Ä Å Æ Ç
fila11 È É Ê Ë Ì Í Î Ï Ð Ñ Ò Ó Ô Õ Ö × Ø Ù Ú Û
fila12 Ü Ý Þ ß à á â ã ä å æ ç è é ê ë ì í î ï
fila13 ð ñ ò ó ô õ ö ÷ ø ù ú û ü ý þ ÿ

*P3D324532.exe*

fila0 □ □ □ □ □ □ □ □ □ □ □ □ □ □
fila1 □ □ □ □ □ □ □ □ □ □ □ - ! " # $ % & '
fila2 ( ) * + , - . / 0 1 2 3 4 5 6 7 8 9 : ;
fila3 < = > ? @ A B C D E F G H I J K L M N O
fila4 P Q R S T U V W X Y Z [ \ ] ^ _ ` a b c
fila5 d e f g h i j k l m n o p q r s t u v w
fila6 x y z { | } ~ □ € □ , *f* „ … † ‡ ˆ ‰ Š ‹
fila7 Œ □ Ž □ □ ‘ ’ “ ” • – — ˜ ™ š › œ □ ž Ÿ
fila8 ¡ ¢ £ ¤ ¥ ¦ § ¨ © ª « ¬ - ® ¯ ° ± ² ³
fila9 ´ µ ¶ · ¸ ¹ º » ¼ ½ ¾ ¿ À Á Â Ã Ä Å Æ Ç
fila10 È É Ê Ë Ì Í Î Ï Ð Ñ Ò Ó Ô Õ Ö × Ø Ù Ú Û
fila11 Ü Ý Þ ß à á

*P3D34354567.exe*

Fila 1 □ □ □ □ □ □ □ □ □ □ □ □ □ □
Fila 2 □□ □ □ □ □ □ □ □ □ - ! " # $ % & ' (
Fila 3 () * + , - . / 0 1 2 3 4 5 6 7 8 9 : ; <
Fila 4 <= > ? @ A B C D E F G H I J K L M N O P
Fila 5 PQ R S T U V W X Y Z [ \ ] ^ _ ` a b c d
Fila 6 de f g h i j k l m n o p q r s t u v w x
Fila 7 xy z { | } ~ □ € □ , *f* „ … † ‡ ˆ ‰ Š ‹ Œ
Fila 8 Œ□ Ž □ □ ‘ ’ “ ” • – — ˜ ™ š › œ □ ž Ÿ
Fila 9 ¡ ¢ £ ¤ ¥ ¦ § ¨ © ª « ¬ - ® ¯ ° ± ² ³ ´
Fila 10 ´µ ¶ · ¸ ¹ º » ¼ ½ ¾ ¿ À Á Â Ã Ä Å Æ Ç È
Fila 11 ÈÉ Ê Ë Ì Í Î Ï Ð Ñ Ò Ó Ô Õ Ö × Ø Ù Ú Û Ü
Fila 12 ÜÝ Þ ß à á â ã ä å æ ç è é ê ë ì í î ï ð
Fila 13 ðñ ò ó ô õ ö ÷ ø ù ú û ü ý þ ÿ

*P3D346778.exe*

```
Fila 1  □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □
fila 2 □ □ □ □ □ □ □ □ □ □ -  ! " # $ % & ' ( )
fila 3 * + , - . / 0 1 2 3 4 5 6 7 8 9 : ; < = >
fila 4 ? @ A B C D E F G H I J K L M N O P Q R S
fila 5 T U V W X Y Z [ \ ] ^ _ ` a b c d e f g h
fila 6 i j k l m n o p q r s t u v w x y z { | }
fila 7 ~ □ € □ , ƒ „ … † ‡ ˆ ‰ Š ‹ Œ □ Ž □ □ ‘ ’
fila 8 “ ” • – — ˜ ™ š › œ □ ž Ÿ  ¡ ¢ £ ¤ ¥ ¦ §
fila 9 ¨ © ª « ¬ - ® ¯ ° ± ² ³ ´ µ ¶ · ¸ ¹ º » ¼
fila 10 ½ ¾ ¿ À Á Â Ã Ä Å Æ Ç È É Ê Ë Ì Í Î Ï Ð Ñ
fila 11 Ò Ó Ô Õ Ö × Ø Ù Ú Û Ü Ý Þ ß à á â ã ä å æ
fila 12 ç è é ê ë ì í î ï ð ñ ò ó ô õ ö ÷ ø ù ú û
fila 13 ü ý þ ÿ
```

*P3D4564567.exe*

```
fila1 □ □ □ □ □ □ □ □ □ □ □ □ □ □
fila2 □ □ □ □ □ □ □ □ □ □ □ -  ! " # $ % & '
fila3 ( ) * + , - . / 0 1 2 3 4 5 6 7 8 9 : ;
fila4 < = > ? @ A B C D E F G H I J K L M N O
fila5 P Q R S T U V W X Y Z [ \ ] ^ _ ` a b c
fila6 d e f g h i j k l m n o p q r s t u v w
fila7 x y z { | } ~ □ € □ , ƒ „ … † ‡ ˆ ‰ Š ‹
fila8 Œ □ Ž □ □ ‘ ’ “ ” • – — ˜ ™ š › œ □ ž Ÿ
fila9  ¡ ¢ £ ¤ ¥ ¦ § ¨ © ª « ¬ - ® ¯ ° ± ² ³
fila10 ´ µ ¶ · ¸ ¹ º » ¼ ½ ¾ ¿ À Á Â Ã Ä Å Æ Ç
fila11 È É Ê Ë Ì Í Î Ï Ð Ñ Ò Ó Ô Õ Ö × Ø Ù Ú Û
fila12 Ü Ý Þ ß à á â ã ä å æ ç è é ê ë ì í î ï
fila13 ð ñ ò ó ô õ ö ÷ ø ù ú û ü ý þ ÿ
```

*P3D54546765.exe*

```
fila1  □ □ □ □ □ □ □ □ □ □ □ □ □ □
fila2 □ □ □ □ □ □ □ □ □ □ □ -  ! " # $ % & '
fila3 ( ) * + , - . / 0 1 2 3 4 5 6 7 8 9 : ;
fila4 < = > ? @ A B C D E F G H I J K L M N O
fila5 P Q R S T U V W X Y Z [ \ ] ^ _ ` a b c
fila6 d e f g h i j k l m n o p q r s t u v w
fila7 x y z { | } ~ □ € □ , ƒ „ … † ‡ ˆ ‰ Š ‹
fila8 Œ □ Ž □ □ ‘ ’ “ ” • – — ˜ ™ š › œ □ ž Ÿ
fila9  ¡ ¢ £ ¤ ¥ ¦ § ¨ © ª « ¬ - ® ¯ ° ± ² ³
fila10 ´ µ ¶ · ¸ ¹ º » ¼ ½ ¾ ¿ À Á Â Ã Ä Å Æ Ç
fila11 È É Ê Ë Ì Í Î Ï Ð Ñ Ò Ó Ô Õ Ö × Ø Ù Ú Û
fila12 Ü Ý Þ ß à á â ã ä å æ ç è é ê ë ì í î ï
fila13 ð ñ ò ó ô õ ö ÷ ø ù ú û ü ý þ ÿ
```

*P3D56347845.exe*

```
Fila 1   □ □ □ □ □ □ □ □ □ □ □ □ □ □
Fila 2 □ □ □ □ □ □ □ □ □ □ -    ! " # $ % & '
Fila 3 ( ) * + , - . / 0 1 2 3 4 5 6 7 8 9 : ;
Fila 4 < = > ? @ A B C D E F G H I J K L M N O
Fila 5 P Q R S T U V W X Y Z [ \ ] ^ _ ` a b c
Fila 6 d e f g h i j k l m n o p q r s t u v w
Fila 7 x y z { | } ~ □ € □ , ƒ „ … † ‡ ˆ ‰ Š ‹
Fila 8 Œ □ Ž □ □ ‘ ’ “ ” • – — ˜ ™ š › œ □ ž Ÿ
Fila 9   ¡ ¢ £ ¤ ¥ ¦ § ¨ © ª « ¬ - ® ¯ ° ± ² ³
Fila 10 ´ µ ¶ · ¸ ¹ º » ¼ ½ ¾ ¿ À Á Â Ã Ä Å Æ Ç
Fila 11 È É Ê Ë Ì Í Î Ï Ð Ñ Ò Ó Ô Õ Ö × Ø Ù Ú Û
Fila 12 Ü Ý Þ ß à á â ã ä å æ ç è é ê ë ì í î ï
Fila 13 ð ñ ò ó ô õ ö ÷ ø ù ú û ü ý þ ÿ
```

*P3PROF.exe*

```
Fila 1.   □ □ □ □ □ □ □ □ □ □ □ □ □ □
Fila 2.   □ □ □ □ □ □ □ □ □ □ -    ! " # $ % & ' (
Fila 3.  ) * + , - . / 0 1 2 3 4 5 6 7 8 9 : ; <
Fila 4.  = > ? @ A B C D E F G H I J K L M N O P
Fila 5.  Q R S T U V W X Y Z [ \ ] ^ _ ` a b c d
Fila 6.  e f g h i j k l m n o p q r s t u v w x
Fila 7.  y z { | } ~ □ € □ , ƒ „ … † ‡ ˆ ‰ Š ‹ Œ
Fila 8.  □ Ž □ □ ‘ ’ “ ” • – — ˜ ™ š › œ □ ž Ÿ
Fila 9.  ¡ ¢ £ ¤ ¥ ¦ § ¨ © ª « ¬ - ® ¯ ° ± ² ³ ´
Fila 10.  µ ¶ · ¸ ¹ º » ¼ ½ ¾ ¿ À Á Â Ã Ä Å Æ Ç È
Fila 11.  É Ê Ë Ì Í Î Ï Ð Ñ Ò Ó Ô Õ Ö × Ø Ù Ú Û Ü
Fila 12.  Ý Þ ß à á â ã ä å æ ç è é ê ë ì í î ï ð
Fila 13.  ñ ò ó ô õ ö ÷ ø ù ú û ü ý þ ÿ
```

### 6.1.4.- Comparación de los ejecutables dejados y los generados

Se compararon los resultados de la ejecución de los *.exe* disponibles con los de la ejecución de los *.exe* generados tras la compilación de las prácticas. Los resultados obtenidos son los siguientes:

```
-- RESULTADO DE LA COMPARACIÓN DE LOS .EXE  --

--------------------------------------------------------------------------------
Nombre de la práctica       Tiene exe     Genera exe     Comparación

P3D1347678.pas                  SI              SI

- Entradas en el directorio C:\arma\Ascii\GA_Entradas\ y comienzan por: GA1
      - Fichero de salida: SALIDA.TXT
                                                                    IGUALES

--------------------------------------------------------------------------------
Nombre de la práctica       Tiene exe     Genera exe     Comparación

P3D1354647.pas                  SI              SI
```

- Entradas en el directorio C:\arma\Ascii\GA_Entradas\ y comienzan por: GA1
    - Fichero de salida: SALIDA.TXT

IGUALES

---

Nombre de la práctica Tiene exe Genera exe Comparación

P3D1425678.pas SI SI

- Entradas en el directorio C:\arma\Ascii\GA_Entradas\ y comienzan por: GA1
    - Fichero de salida: SALIDA.TXT

IGUALES

---

Nombre de la práctica Tiene exe Genera exe Comparación

P3D1456787.pas SI SI

- Entradas en el directorio C:\arma\Ascii\GA_Entradas\ y comienzan por: GA1
    - Fichero de salida: SALIDA.TXT

IGUALES

---

Nombre de la práctica Tiene exe Genera exe Comparación

P3D213564.pas SI SI

- Entradas en el directorio C:\arma\Ascii\GA_Entradas\ y comienzan por: GA1
    - Fichero de salida: SALIDA.TXT

IGUALES

---

Nombre de la práctica Tiene exe Genera exe Comparación

P3D23432627.pas SI SI

- Entradas en el directorio C:\arma\Ascii\GA_Entradas\ y comienzan por: GA1
    - Fichero de salida: SALIDA.TXT

IGUALES

---

Nombre de la práctica Tiene exe Genera exe Comparación

P3D324532.pas SI SI

- Entradas en el directorio C:\arma\Ascii\GA_Entradas\ y comienzan por: GA1
    - Fichero de salida: SALIDA.TXT

IGUALES

---

Nombre de la práctica Tiene exe Genera exe Comparación

P3D34354567.pas SI SI

- Entradas en el directorio C:\arma\Ascii\GA_Entradas\ y comienzan por: GA1

```
    - Fichero de salida: S1.TXT
                                                              IGUALES

-------------------------------------------------------------------------

Nombre de la práctica      Tiene exe    Genera exe    Comparación

P3D346778.pas                  SI             SI

- Entradas en el directorio C:\arma\Ascii\GA_Entradas\ y comienzan por: GA1
    - Fichero de salida: SALIDA.TXT
                                                              IGUALES

-------------------------------------------------------------------------
Nombre de la práctica      Tiene exe    Genera exe    Comparación

P3D4564567.pas                 SI             SI

- Entradas en el directorio C:\arma\Ascii\GA_Entradas\ y comienzan por: GA1
    - Fichero de salida: S1.TXT
                                                              IGUALES

-------------------------------------------------------------------------
Nombre de la práctica      Tiene exe    Genera exe    Comparación

P3D54546765.pas                SI             SI

- Entradas en el directorio C:\arma\Ascii\GA_Entradas\ y comienzan por: GA1
    - Fichero de salida: S1.TXT
                                                              IGUALES

-------------------------------------------------------------------------
Nombre de la práctica      Tiene exe    Genera exe    Comparación

P3D56347845.pas                SI             SI

- Entradas en el directorio C:\arma\Ascii\GA_Entradas\ y comienzan por: GA1
    - Fichero de salida: S1.TXT
                                                              IGUALES
```

En todas las prácticas disponibles, la ejecución del *.exe* dejado y el generado coinciden, como se puede observar en los resultados obtenidos.

### 6.1.5.- Comparación de los ejecutables con la práctica del profesor

Se compararon la ejecución del *.exe* del alumno, con la ejecución del *.exe* del profesor. Los resultados obtenidos son los siguientes:

```
--  RESULTADO  DE  LA  COMPARACIÓN  DE  LOS  .EXE  CON  LA  PRÁCTICA  DEL
PROFESOR  --


-------------------------------------------------------------------------
```

Nombre de la práctica Tiene exe PRAC. PROF Comparación

P3D1347678.pas SI SI

- Entradas en el directorio C:\arma\Ascii\GA_Entradas\ y comienzan por: GA1
  - Fichero de salida: SALIDA.TXT

IGUALES

---

Nombre de la práctica Tiene exe PRAC. PROF Comparación

P3D1354647.pas SI SI

- Entradas en el directorio C:\arma\Ascii\GA_Entradas\ y comienzan por: GA1
  - Fichero de salida: SALIDA.TXT

DISTINTOS

---

Nombre de la práctica Tiene exe PRAC. PROF Comparación

P3D1425678.pas SI SI

- Entradas en el directorio C:\arma\Ascii\GA_Entradas\ y comienzan por: GA1
  - Fichero de salida: SALIDA.TXT

DISTINTOS

---

Nombre de la práctica Tiene exe PRAC. PROF Comparación

P3D1456787.pas SI SI

- Entradas en el directorio C:\arma\Ascii\GA_Entradas\ y comienzan por: GA1
  - Fichero de salida: SALIDA.TXT

DISTINTOS

---

Nombre de la práctica Tiene exe PRAC. PROF Comparación

P3D213564.pas SI SI

- Entradas en el directorio C:\arma\Ascii\GA_Entradas\ y comienzan por: GA1
  - Fichero de salida: SALIDA.TXT

DISTINTOS

---

Nombre de la práctica Tiene exe PRAC. PROF Comparación

P3D23432627.pas SI SI

- Entradas en el directorio C:\arma\Ascii\GA_Entradas\ y comienzan por: GA1
  - Fichero de salida: SALIDA.TXT

DISTINTOS

---

Nombre de la práctica Tiene exe PRAC. PROF Comparación

P3D324532.pas SI SI

- Entradas en el directorio C:\arma\Ascii\GA_Entradas\ y comienzan por: GA1
- Fichero de salida: SALIDA.TXT

DISTINTOS

---

Nombre de la práctica Tiene exe PRAC. PROF Comparación

P3D34354567.pas SI SI

- Entradas en el directorio C:\arma\Ascii\GA_Entradas\ y comienzan por: GA1
- Fichero de salida: IMPOSIBLE IDENTIFICAR LAS SALIDAS. NOMBRE ERRÓNEO EN LA PRÁCTICA

DISTINTOS

---

Nombre de la práctica Tiene exe PRAC. PROF Comparación

P3D346778.pas SI SI

- Entradas en el directorio C:\arma\Ascii\GA_Entradas\ y comienzan por: GA1
- Fichero de salida: SALIDA.TXT

DISTINTOS

---

Nombre de la práctica Tiene exe PRAC. PROF Comparación

P3D4564567.pas SI SI

- Entradas en el directorio C:\arma\Ascii\GA_Entradas\ y comienzan por: GA1
- Fichero de salida: IMPOSIBLE IDENTIFICAR LAS SALIDAS. NOMBRE ERRÓNEO EN LA PRÁCTICA

DISTINTOS

---

Nombre de la práctica Tiene exe PRAC. PROF Comparación

P3D54546765.pas SI SI

- Entradas en el directorio C:\arma\Ascii\GA_Entradas\ y comienzan por: GA1
- Fichero de salida: IMPOSIBLE IDENTIFICAR LAS SALIDAS. NOMBRE ERRÓNEO EN LA PRÁCTICA

DISTINTOS

---

Nombre de la práctica Tiene exe PRAC. PROF Comparación

P3D56347845.pas SI SI

- Entradas en el directorio C:\arma\Ascii\GA_Entradas\ y comienzan por: GA1
- Fichero de salida: IMPOSIBLE IDENTIFICAR LAS SALIDAS. NOMBRE ERRÓNEO EN LA PRÁCTICA

DISTINTOS

A la vista de los resultados, se puede observar que de doce prácticas, sólo la salida de una coincide con la salida del profesor. Eso puede ser debido, a que al comparar las salidas de los alumnos con las del profesor, los archivos se comparan línea a línea, y es posible que la ejecución sea correcta, pero que los datos de salida no estén escritos de la misma forma que los del profesor.

De las once prácticas cuyas salidas son distintas que las del profesor, siete son distintas por el motivo citado anteriormente. El resto son distintas porque el nombre del archivo de salida no es el mismo que el del archivo del profesor.

### 6.1.6.- Comparación de los ejecutables generados con la práctica del profesor

Los resultados obtenidos son los siguientes:

```
-- RESULTADO DE LA COMPARACIÓN DE LOS .EXE GENERADOS CON LA PRÁCTICA DEL PROFESOR --

--------------------------------------------------------------------------------
Nombre de la práctica       Tiene exe    PRAC. PROF    Comparación

P3D1347678.pas                  SI             SI

- Entradas en el directorio C:\arma\Ascii\GA_Entradas\ y comienzan por: GA1
     - Fichero de salida: SALIDA.TXT
                                                                    IGUALES

--------------------------------------------------------------------------------
Nombre de la práctica       Tiene exe    PRAC. PROF    Comparación

P3D1354647.pas                  SI             SI

- Entradas en el directorio C:\arma\Ascii\GA_Entradas\ y comienzan por: GA1
     - Fichero de salida: SALIDA.TXT
                                                          DISTINTOS

--------------------------------------------------------------------------------
Nombre de la práctica       Tiene exe    PRAC. PROF    Comparación

P3D1425678.pas                  SI             SI

- Entradas en el directorio C:\arma\Ascii\GA_Entradas\ y comienzan por: GA1
     - Fichero de salida: SALIDA.TXT
                                                                    DISTINTOS

--------------------------------------------------------------------------------
Nombre de la práctica       Tiene exe    PRAC. PROF    Comparación

P3D1456787.pas                  SI             SI

- Entradas en el directorio C:\arma\Ascii\GA_Entradas\ y comienzan por: GA1
     - Fichero de salida: SALIDA.TXT
                                                          DISTINTOS
```

```
------------------------------------------------------------------------
Nombre de la práctica       Tiene exe     PRAC. PROF     Comparación

P3D213564.pas                    SI              SI

- Entradas en el directorio C:\arma\Ascii\GA_Entradas\ y comienzan por: GA1
      - Fichero de salida: SALIDA.TXT
                                                                  DISTINTOS

------------------------------------------------------------------------
Nombre de la práctica       Tiene exe     PRAC. PROF     Comparación

P3D23432627.pas                  SI              SI

- Entradas en el directorio C:\arma\Ascii\GA_Entradas\ y comienzan por: GA1
      - Fichero de salida: SALIDA.TXT
                                                          DISTINTOS

------------------------------------------------------------------------
Nombre de la práctica       Tiene exe     PRAC. PROF     Comparación

P3D324532.pas                    SI              SI

- Entradas en el directorio C:\arma\Ascii\GA_Entradas\ y comienzan por: GA1
      - Fichero de salida: SALIDA.TXT
                                                          DISTINTOS

------------------------------------------------------------------------
Nombre de la práctica       Tiene exe     PRAC. PROF     Comparación

P3D34354567.pas                  SI              SI

- Entradas en el directorio C:\arma\Ascii\GA_Entradas\ y comienzan por: GA1
      - Fichero de salida: IMPOSIBLE IDENTIFICAR LAS SALIDAS. NOMBRE ERRÓNEO
EN LA PRÁCTICA
                                                                  DISTINTOS

------------------------------------------------------------------------
Nombre de la práctica       Tiene exe     PRAC. PROF     Comparación

P3D346778.pas                    SI              SI

- Entradas en el directorio C:\arma\Ascii\GA_Entradas\ y comienzan por: GA1
      - Fichero de salida: SALIDA.TXT
                                                          DISTINTOS

------------------------------------------------------------------------
Nombre de la práctica       Tiene exe     PRAC. PROF     Comparación

P3D4564567.pas                   SI              SI

- Entradas en el directorio C:\arma\Ascii\GA_Entradas\ y comienzan por: GA1
      - Fichero de salida: IMPOSIBLE IDENTIFICAR LAS SALIDAS. NOMBRE ERRÓNEO
EN LA PRÁCTICA
```

```
                                                                    DISTINTOS

--------------------------------------------------------------------------------
Nombre de la práctica       Tiene exe     PRAC. PROF     Comparación

P3D54546765.pas                 SI              SI

- Entradas en el directorio C:\arma\Ascii\GA_Entradas\ y comienzan por: GA1
     - Fichero de salida: IMPOSIBLE IDENTIFICAR LAS SALIDAS. NOMBRE ERRÓNEO
EN LA PRÁCTICA
                                                                    DISTINTOS

--------------------------------------------------------------------------------
Nombre de la práctica       Tiene exe     PRAC. PROF     Comparación

P3D56347845.pas                 SI              SI

- Entradas en el directorio C:\arma\Ascii\GA_Entradas\ y comienzan por: GA1
     - Fichero de salida: IMPOSIBLE IDENTIFICAR LAS SALIDAS. NOMBRE ERRÓNEO
EN LA PRÁCTICA
                                                                    DISTINTOS
```

Los resultados obtenidos son idénticos a los resultados del apartado anterior, ya que como hemos visto anteriormente, la ejecución de los *.exe* de los alumnos es igual que la de los *.exe* generados tras la compilación de los programas.

### 6.1.7.- Comparación de las estructuras de los programas con la práctica del profesor

Los niveles de aproximación al código del profesor obtenidos son los siguientes:

```
-- LISTADO CON LOS NIVELES DE APROXIMACIÓN  --

Nombre de la práctica     Nivel de aproximación

P3D1347678.pas            |0
P3D1354647.pas            |0
P3D1425678.pas            |0
P3D1456787.pas            |0
P3D213564.pas             |0
P3D23432627.pas           |0
P3D324532.pas             |0
P3D34354567.pas           |0
P3D346778.pas             |0
P3D4564567.pas            |0
P3D54546765.pas           |0
P3D56347845.pas           |0
```

Los niveles son todos cero, ya que la estructura de las prácticas es muy similar.

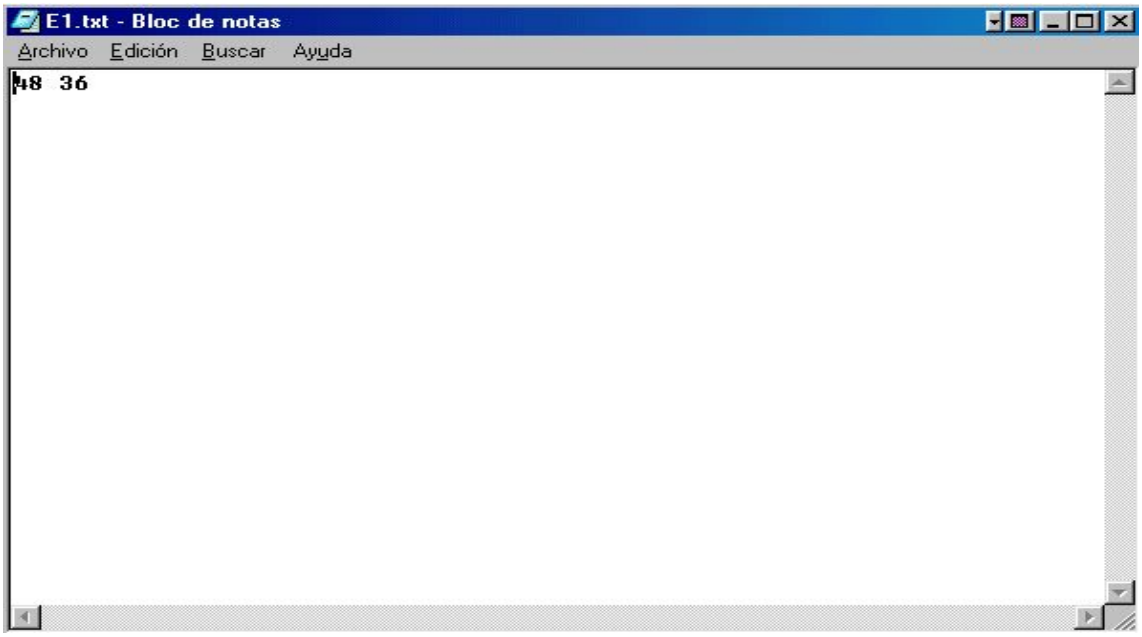
## 6.2.- PRUEBAS DE LA PRÁCTICA DEL CÁLCULO DEL M.C.D.

Para la realización de estas pruebas disponemos de 10 prácticas entregadas por los alumnos. El código de cada una de ellas se puede encontrar en el apéndice I.

Esta práctica consiste en leer dos números que se encuentran en un fichero de entrada de nombre *e1.txt*, calcular el máximo común divisor y escribir el resultado en un fichero de salida de nombre *salida.txt*.

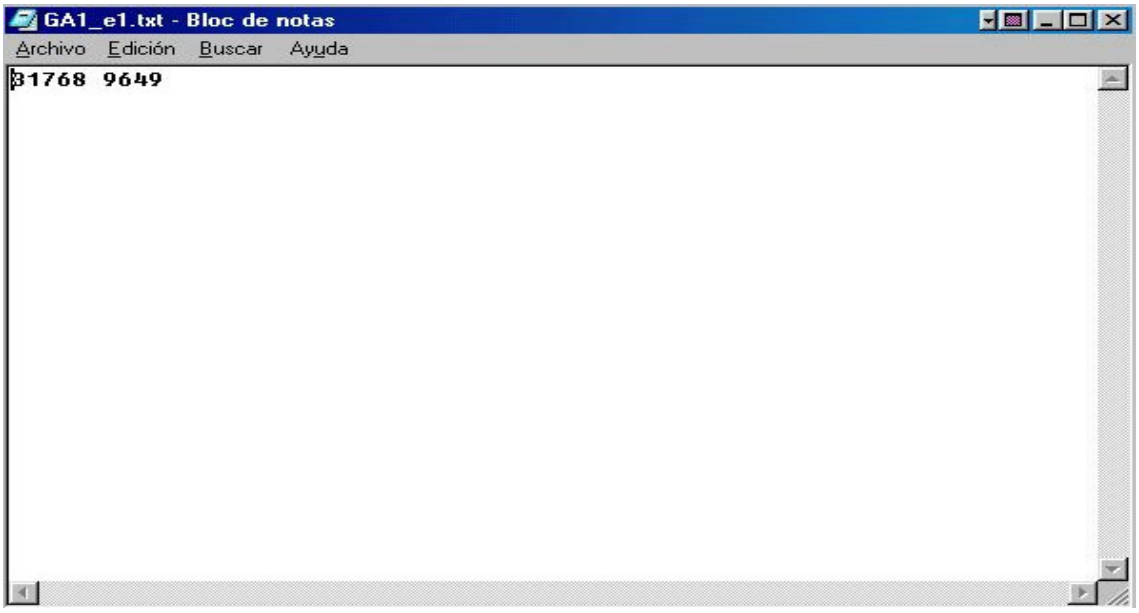
Se han realizado dos tipo de prueba: generación manual y generación automática. Para la generación manual hemos utilizado un fichero de entrada creado por nosotros:

*e1.txt*




Y para la generación automática, la herramienta nos creo un fichero aleatorio con dos números enteros positivos como el siguiente:

*e1.txt*



## 6.2.1.- Generación manual.

### 6.2.1.1.- Ejecución de los *.exe* de los alumnos.

Puesto que para está práctica ningún alumno ha dejado el ejecutable de su práctica esta opción no tiene sentido.

### 6.2.1.2.- Ejecución de la práctica de un único alumno con datos de entrada específicos

Puesto que en la siguiente opción compilaremos y ejecutaremos todas las prácticas con los datos de entrada que nosotros hemos especificado esta opción no tiene sentido.

### 6.2.1.3.- Compilación y ejecución de los programas

Se compilaron las 10 prácticas de los alumnos obteniéndose el siguiente resultado:

```
-- RESULTADO DE LA COMPILACIÓN --

COMPILA: P1D23432627.pas

COMPILA: P1D346778.pas

COMPILA: P1D1347678.pas

COMPILA: P1D1354647.pas

COMPILA: P1D1425678.pas

COMPILA: P1D56347845.pas

COMPILA: P1D34354567.pas

COMPILA: P1D78679540.pas

COMPILA: P1D54546765.pas

COMPILA: P1D213564.pas
```

Es decir compilaron el 100 % de las prácticas entregadas por los alumnos.

Se ejecutaron los diez ejecutables generados tras la compilación produciéndose los siguientes resultados:

```
-- RESULTADO DE LA EJECUCIÓN --


--------------------------------------------------------------------------------
- RESULTADOS PARA LOS DATOS DE ENTRADA QUE ESTÁN EN EL DIRECTORIO
C:\ARMA\Mcd\Compiladas\ExeGenerados\GM_Entradas\GM_Entrada1\

EJECUTA: P1D23432627.EXE
LA SALIDA QUE GENERA LA EJECUCIÓN SE PUEDE ENCONTRAR EN
C:\ARMA\Mcd\Compiladas\ExeGenerados\GM_Salida1\P1D23432627\

EJECUTA: P1D346778.EXE
LA SALIDA QUE GENERA LA EJECUCIÓN SE PUEDE ENCONTRAR EN
C:\ARMA\Mcd\Compiladas\ExeGenerados\GM_Salida1\P1D346778\

EJECUTA: P1D1347678.EXE
LA SALIDA QUE GENERA LA EJECUCIÓN SE PUEDE ENCONTRAR EN
C:\ARMA\Mcd\Compiladas\ExeGenerados\GM_Salida1\P1D1347678\

NO EJECUTA: P1D1354647.EXE

EJECUTA: P1D1425678.EXE
LA SALIDA QUE GENERA LA EJECUCIÓN SE PUEDE ENCONTRAR EN
C:\ARMA\Mcd\Compiladas\ExeGenerados\GM_Salida1\P1D1425678\

EJECUTA: P1D56347845.EXE
LA SALIDA QUE GENERA LA EJECUCIÓN SE PUEDE ENCONTRAR EN
C:\ARMA\Mcd\Compiladas\ExeGenerados\GM_Salida1\P1D56347845\

EJECUTA: P1D34354567.EXE
LA SALIDA QUE GENERA LA EJECUCIÓN SE PUEDE ENCONTRAR EN
C:\ARMA\Mcd\Compiladas\ExeGenerados\GM_Salida1\P1D34354567\

EJECUTA: P1D78679540.EXE
LA SALIDA QUE GENERA LA EJECUCIÓN SE PUEDE ENCONTRAR EN
C:\ARMA\Mcd\Compiladas\ExeGenerados\GM_Salida1\P1D78679540\

EJECUTA: P1D54546765.EXE
LA SALIDA QUE GENERA LA EJECUCIÓN SE PUEDE ENCONTRAR EN
C:\ARMA\Mcd\Compiladas\ExeGenerados\GM_Salida1\P1D54546765\

EJECUTA: P1D213564.EXE
LA SALIDA QUE GENERA LA EJECUCIÓN SE PUEDE ENCONTRAR EN
C:\ARMA\Mcd\Compiladas\ExeGenerados\GM_Salida1\P1D213564\
```

Es decir, se ejecuta el 90 % de las prácticas.

Las salidas que genera cada práctica que sí ejecuta son las siguientes:

*P1D1347678.EXE*

| el maximo comun divisor de 48 y 36 es 12 |
|---|

*P1D1425678.EXE*

| El mcd es: 12 |
|---|

*P1D213564.EXE*

| El maximo comun divisor de  48  36 es =12 |
|---|

*P1D23432627.EXE*

| El M.C.D. de  48  36 es 12 |
|---|

*P1D34354567.EXE*

| 12 |
|---|

*P1D346778.EXE*

| El mcd de 48 y 36 es 12 |
|---|

*P1D54546765.EXE*

| 12 |
|---|

*P1D56347845.EXE*

| 12 |
|---|

*P1D78679540.EXE*

| 12 |
|---|

### 6.2.1.4.- Ejecución del *.exe* y comparación con la compilación y posterior ejecución de dicha práctica

Puesto que para está práctica ningún alumno ha dejado el ejecutable de su práctica esta opción no tiene sentido.

### 6.2.1.5.- Comparación de los resultados con los del  profesor

Puesto que para esta práctica no se disponía de la solución del profesor esta opción no tiene sentido.

### 6.2.1.6.- Comparación de las estructuras de los programas

Puesto que para esta práctica no se disponía de la solución del profesor esta opción no tiene sentido.

## 6.2.2.- Generación automática.

### 6.2.2.1.- Ejecución de los *.exe* de los alumnos

Puesto que para está práctica ningún alumno ha dejado el ejecutable de su práctica esta opción no tiene sentido.

### 6.2.2.2.- Ejecución de la práctica de un único alumno con datos de entrada específicos

Esta opción no esta disponible para la generación automática de los datos.

### 6.2.2.3.- Compilación y ejecución de los programas

Se compilaron las 10 prácticas de los alumnos obteniéndose el siguiente resultado:

```
-- RESULTADO DE LA COMPILACIÓN --

COMPILA: P1D23432627.pas

COMPILA: P1D346778.pas

COMPILA: P1D1347678.pas

COMPILA: P1D1354647.pas

COMPILA: P1D1425678.pas

COMPILA: P1D56347845.pas

COMPILA: P1D34354567.pas

COMPILA: P1D78679540.pas

COMPILA: P1D54546765.pas

COMPILA: P1D213564.pas
```

Es decir compilaron el 100 % de las prácticas entregadas por los alumnos.

Se ejecutaron los diez ejecutables generados tras la compilación produciéndose los siguientes resultados:

```
--  RESULTADO DE LA EJECUCIÓN  --


--------------------------------------------------------------------------------
- RESULTADOS PARA LOS DATOS DE ENTRADA QUE ESTÁN EN EL DIRECTORIO:
C:\ARMA\Mcd\Compiladas\ExeGenerados\GA_Entradas\ Y QUE EMPIEZAN POR GA1

EJECUTA: P1D23432627.EXE
LA SALIDA QUE GENERA LA EJECUCIÓN SE PUEDE ENCONTRAR EN
C:\ARMA\Mcd\Compiladas\ExeGenerados\GA_Salida1\P1D23432627\

EJECUTA: P1D346778.EXE
LA SALIDA QUE GENERA LA EJECUCIÓN SE PUEDE ENCONTRAR EN
C:\ARMA\Mcd\Compiladas\ExeGenerados\GA_Salida1\P1D346778\

EJECUTA: P1D1347678.EXE
LA SALIDA QUE GENERA LA EJECUCIÓN SE PUEDE ENCONTRAR EN
C:\ARMA\Mcd\Compiladas\ExeGenerados\GA_Salida1\P1D1347678\

NO EJECUTA: P1D1354647.EXE

EJECUTA: P1D1425678.EXE
LA SALIDA QUE GENERA LA EJECUCIÓN SE PUEDE ENCONTRAR EN
C:\ARMA\Mcd\Compiladas\ExeGenerados\GA_Salida1\P1D1425678\

EJECUTA: P1D56347845.EXE
LA SALIDA QUE GENERA LA EJECUCIÓN SE PUEDE ENCONTRAR EN
C:\ARMA\Mcd\Compiladas\ExeGenerados\GA_Salida1\P1D56347845\

EJECUTA: P1D34354567.EXE
LA SALIDA QUE GENERA LA EJECUCIÓN SE PUEDE ENCONTRAR EN
C:\ARMA\Mcd\Compiladas\ExeGenerados\GA_Salida1\P1D34354567\

EJECUTA: P1D78679540.EXE
LA SALIDA QUE GENERA LA EJECUCIÓN SE PUEDE ENCONTRAR EN
C:\ARMA\Mcd\Compiladas\ExeGenerados\GA_Salida1\P1D78679540\

EJECUTA: P1D54546765.EXE
LA SALIDA QUE GENERA LA EJECUCIÓN SE PUEDE ENCONTRAR EN
C:\ARMA\Mcd\Compiladas\ExeGenerados\GA_Salida1\P1D54546765\

EJECUTA: P1D213564.EXE
LA SALIDA QUE GENERA LA EJECUCIÓN SE PUEDE ENCONTRAR EN
C:\ARMA\Mcd\Compiladas\ExeGenerados\GA_Salida1\P1D213564\
```

Es decir, se ejecuta el 90 % de las prácticas.

Las salidas que genera cada práctica que sí ejecuta son las siguientes:

*P1D1347678.EXE*

| el maximo comun divisor de 31768 y 9649 es 1 |
|---|

*P1D1425678.EXE*

| El mcd es: 1 |
|---|

*P1D213564.EXE*

| El maximo comun divisor de31768 9649 es =1 |
|---|

*P1D23432627.EXE*

| El M.C.D. de 31768 9649 es 1 |
|---|

*P1D34354567.EXE*

| 1 |
|---|

*P1D346778.EXE*

| El mcd de 31768 y 9649 es 1 |
|---|

*P1D54546765.EXE*

| 1 |
|---|

*P1D56347845.EXE*

| 1 |
|---|

*P1D78679540.EXE*

| 1 |
|---|

### 6.2.2.4.- Ejecución del *.exe* y comparación con la compilación y posterior ejecución de dicha práctica

Puesto que para está práctica ningún alumno ha dejado el ejecutable de su práctica esta opción no tiene sentido.

### 6.2.2.5.- Comparación de los resultados con los del profesor

Puesto que para esta práctica no se disponía de la solución del profesor esta opción no tiene sentido.

### 6.2.2.6.- Comparación de las estructuras de los programas

Puesto que para esta práctica no se disponía de la solución del profesor esta opción no tiene sentido.

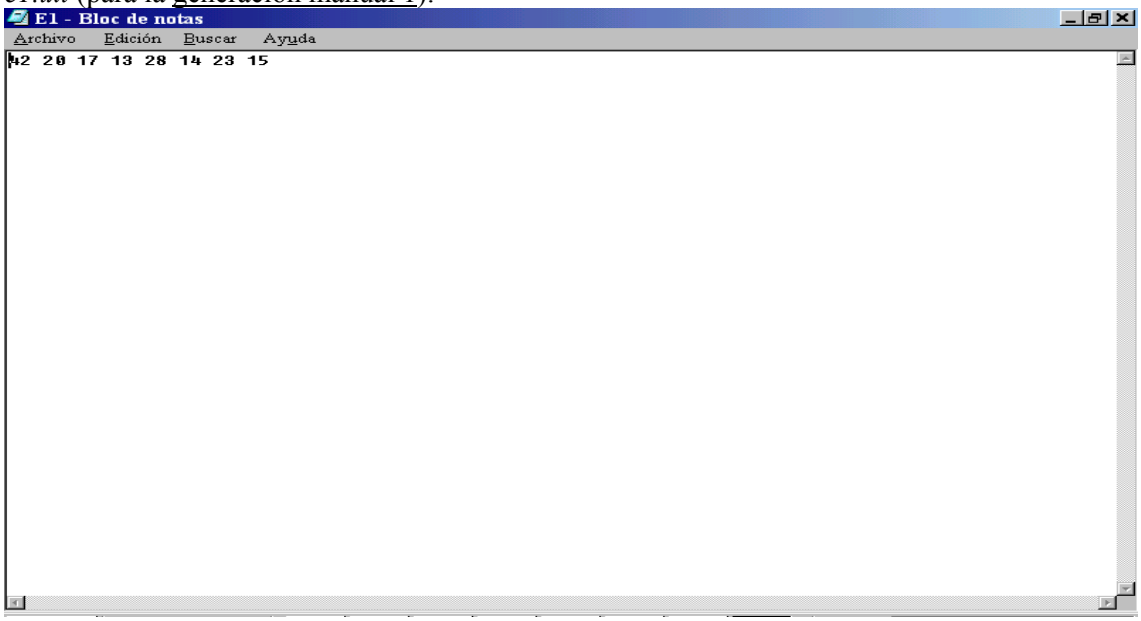
## 6.3.- PRUEBAS DE LA PRÁCTICA DE ORDENACIÓN POR EL MÉTODO DE LA BURBUJA

Para la realización de estas pruebas disponemos de 3 prácticas entregadas por los alumnos. El código de cada una de ellas se puede encontrar en el apéndice I.

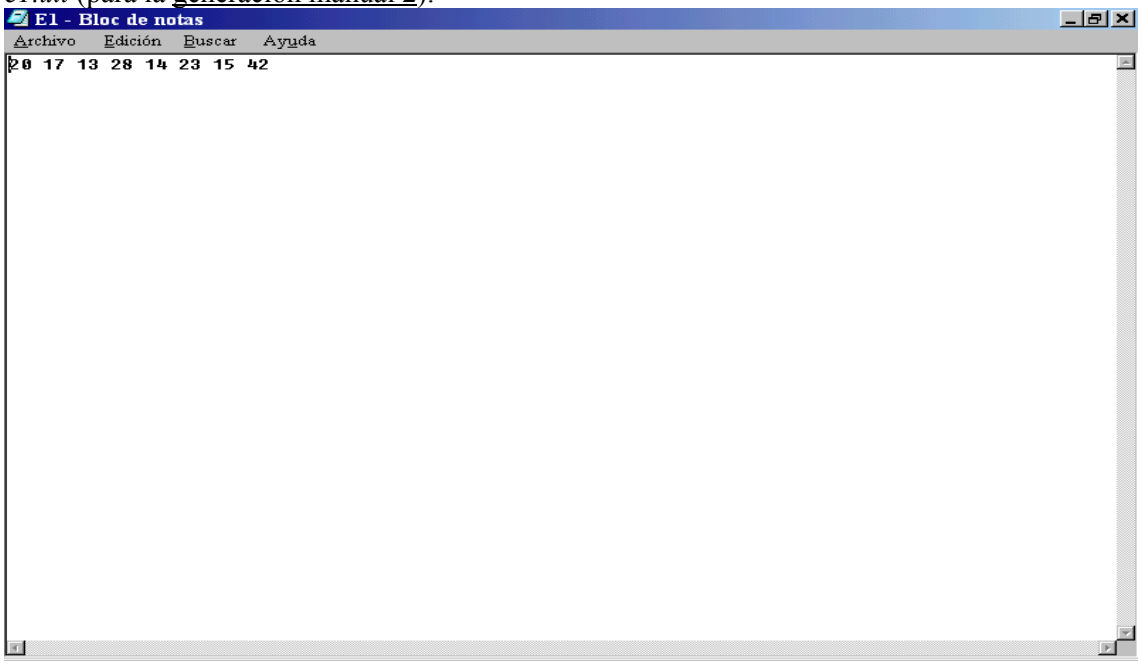
Esta práctica consiste en leer una serie de números enteros que se encuentran en un fichero de entrada de nombre *e1.txt*, ordenar dichos números de menor a mayor (utilizando el método de 'la burbuja') y escribir el resultado en un fichero de salida de nombre *s1.txt*.

Se han realizado dos tipo de prueba: generación manual y generación automática. Para la generación manual hemos utilizado ficheros de entrada creados por nosotros, con ocho elementos. La cantidad de generaciones manuales serán tres, de manera que se tienen los tres ficheros siguientes:

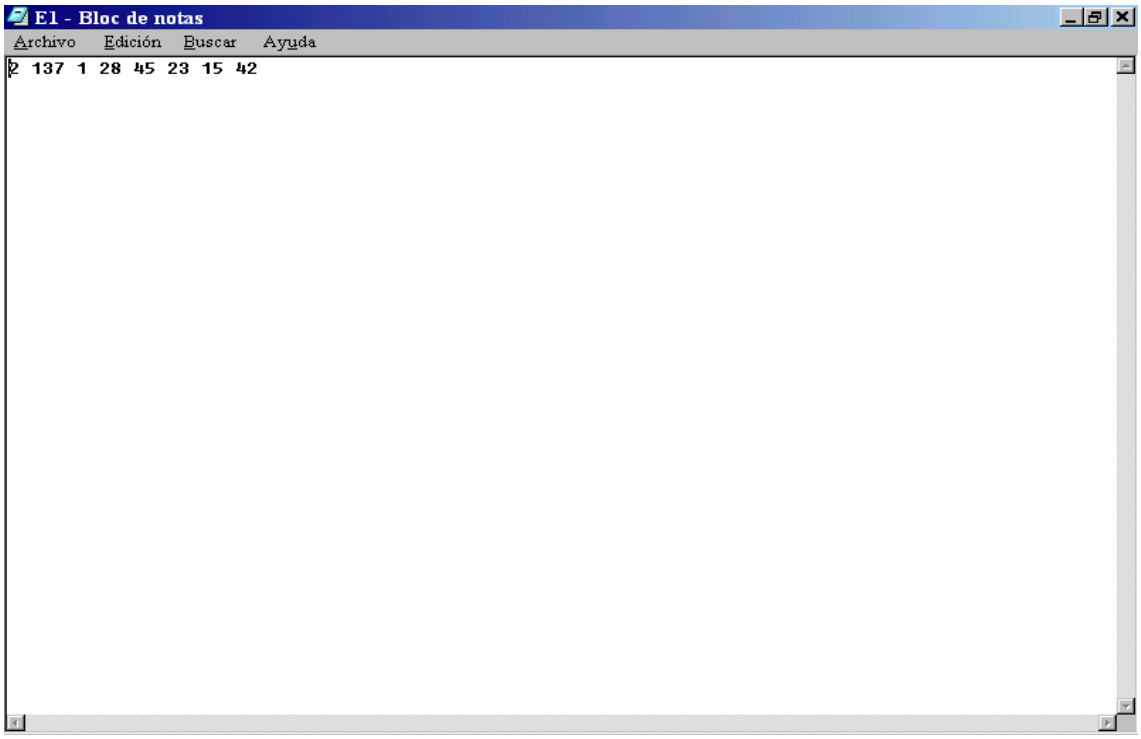
*e1.txt* (para la generación manual 1):




*e1.txt* (para la generación manual 2):

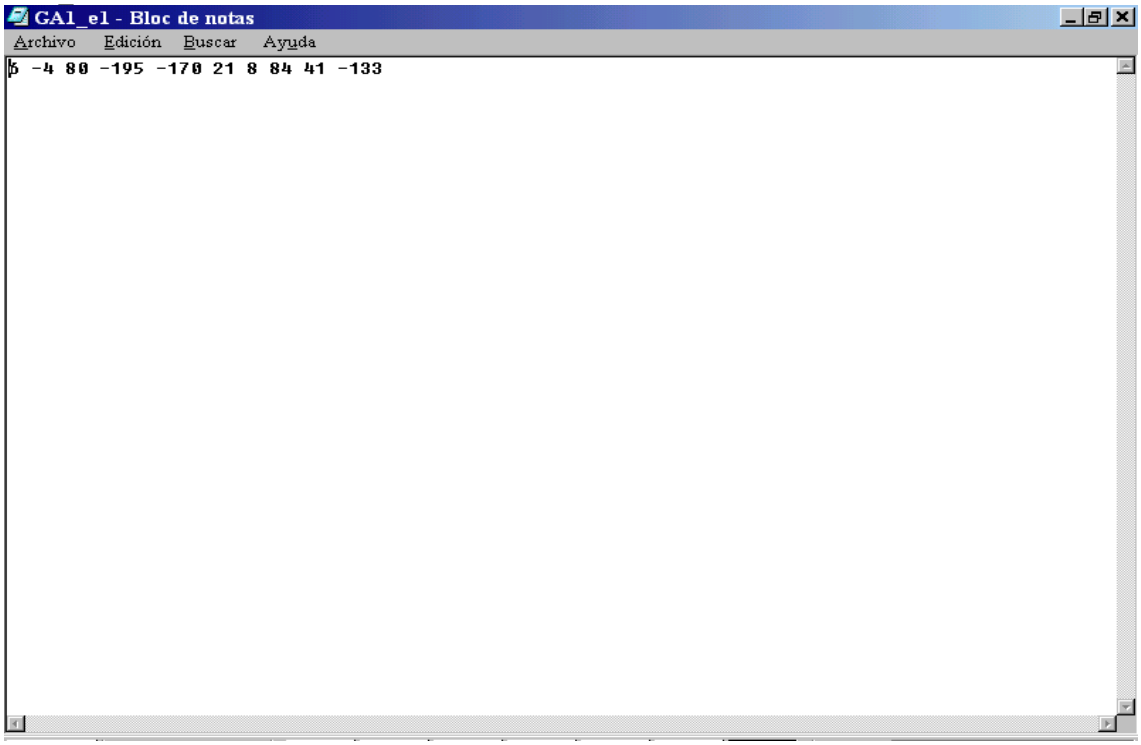


*e1.txt* (para la generación manual 3):

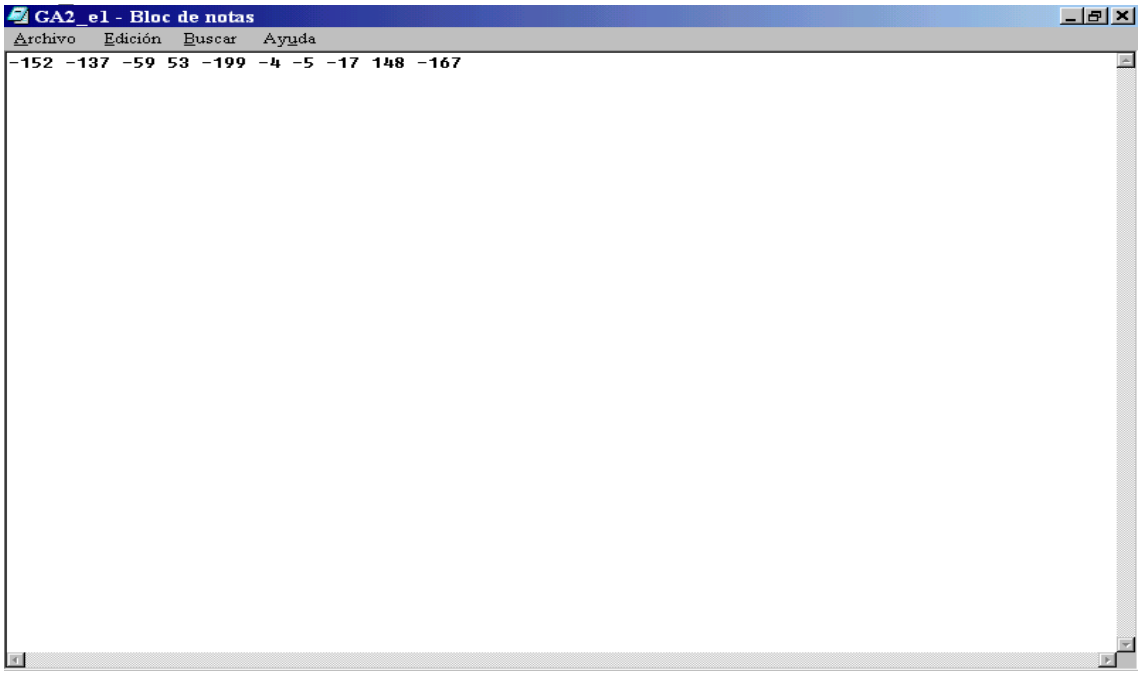



Para la generación automática, crearemos ficheros con diez números comprendidos en el rango [–200,200] (para que haya más diversidad de pruebas), y elegiremos que la cantidad de generaciones automáticas a realizar sean 2. Los ficheros generados han sido los siguientes:

*GA1_e1.txt*



*GA2_e1.txt*




## 6.3.1.- Generación manual.

### 6.3.1.1.- Ejecución de los *.exe* de los almumnos

Puesto que para está práctica ningún alumno ha dejado el ejecutable de su práctica esta opción no tiene sentido.

### 6.3.1.2.- Ejecución de la práctica de un único alumno con datos de entrada específicos

Puesto que en la siguiente opción compilaremos y ejecutaremos todas las prácticas con los datos de entrada que nosotros hemos especificado esta opción no tiene sentido.

### 6.3.1.3.- Compilación y ejecución de los programas

Se compilaron las 3 prácticas de los alumnos obteniéndose el siguiente resultado:

```
-- RESULTADO COMPILACIÓN --

COMPILA: P2D56347845.pas

COMPILA: P2D34354567.pas

COMPILA: P2D4564567.pas
```

Es decir compilaron el 100 % de las prácticas entregadas por los alumnos.

Se ejecutaron los tres ejecutables generados tras la compilación produciéndose los siguientes resultados:

```
-- RESULTADO EJECUCIÓN --

--------------------------------------------------------------------------------
- RESULTADOS PARA LOS DATOS DE ENTRADA QUE ESTÁN EN EL DIRECTORIO
C:\arma\Burbuja\Compiladas\ExeGenerados\GM_Entradas\GM_Entrada1\

EJECUTA: P2D34354567.EXE
LA SALIDA QUE GENERA LA EJECUCIÓN SE PUEDE ENCONTRAR EN
C:\arma\Burbuja\Compiladas\ExeGenerados\GM_Salida1\P2D34354567\

EJECUTA: P2D4564567.EXE
LA SALIDA QUE GENERA LA EJECUCIÓN SE PUEDE ENCONTRAR EN
C:\arma\Burbuja\Compiladas\ExeGenerados\GM_Salida1\P2D4564567\

EJECUTA: P2D56347845.EXE
LA SALIDA QUE GENERA LA EJECUCIÓN SE PUEDE ENCONTRAR EN
C:\arma\Burbuja\Compiladas\ExeGenerados\GM_Salida1\P2D56347845\


--------------------------------------------------------------------------------
- RESULTADOS PARA LOS DATOS DE ENTRADA QUE ESTÁN EN EL DIRECTORIO
C:\arma\Burbuja\Compiladas\ExeGenerados\GM_Entradas\GM_Entrada2\

EJECUTA: P2D34354567.EXE
LA SALIDA QUE GENERA LA EJECUCIÓN SE PUEDE ENCONTRAR EN
C:\arma\Burbuja\Compiladas\ExeGenerados\GM_Salida2\P2D34354567\

EJECUTA: P2D4564567.EXE
LA SALIDA QUE GENERA LA EJECUCIÓN SE PUEDE ENCONTRAR EN
C:\arma\Burbuja\Compiladas\ExeGenerados\GM_Salida2\P2D4564567\

EJECUTA: P2D56347845.EXE
LA SALIDA QUE GENERA LA EJECUCIÓN SE PUEDE ENCONTRAR EN
C:\arma\Burbuja\Compiladas\ExeGenerados\GM_Salida2\P2D56347845\


--------------------------------------------------------------------------------
- RESULTADOS PARA LOS DATOS DE ENTRADA QUE ESTÁN EN EL DIRECTORIO
C:\arma\Burbuja\Compiladas\ExeGenerados\GM_Entradas\GM_Entrada3\

EJECUTA: P2D34354567.EXE
LA SALIDA QUE GENERA LA EJECUCIÓN SE PUEDE ENCONTRAR EN
C:\arma\Burbuja\Compiladas\ExeGenerados\GM_Salida3\P2D34354567\

EJECUTA: P2D4564567.EXE
LA SALIDA QUE GENERA LA EJECUCIÓN SE PUEDE ENCONTRAR EN
C:\arma\Burbuja\Compiladas\ExeGenerados\GM_Salida3\P2D4564567\

EJECUTA: P2D56347845.EXE
LA SALIDA QUE GENERA LA EJECUCIÓN SE PUEDE ENCONTRAR EN
C:\arma\Burbuja\Compiladas\ExeGenerados\GM_Salida3\P2D56347845\
```

Es decir, ejecutan el 100 % de las prácticas.

Las salidas que genera cada práctica que sí ejecuta son las siguientes:

- Para las entradas de generación manual 1:

*P2D34354567.EXE*

```
Se mostrara el vector al principio y en cada vuelta del bucle
externo del metodo de ordenacion
42 20 17 13 28 14 23 15
13 42 20 17 14 28 15 23
13 14 42 20 17 15 28 23
13 14 15 42 20 17 23 28
13 14 15 17 42 20 23 28
13 14 15 17 20 42 23 28
13 14 15 17 20 23 42 28
13 14 15 17 20 23 28 42
13 14 15 17 20 23 28 42
```

*P2D4564567.EXE*

```
0
0
0
0
0
0
0
0
0
0
0
0
0
13
14
15
17
20
23
28
42
```

*P2D56347845.EXE*

```
0 15 14 23 13 17 20 28 2 □ 0 14 15 13 17 20 23 2 28 □ 0 14 13 15 17 20 2 23 28 □ 0
13 14 15 17 2 20 23 28 □ 0 13 14 15 2 17 20 23 28 □ 0 13 14 2 15 17 20 23 28 □ 0 13
2 14 15 17 20 23 28 □ 0 2 13 14 15 17 20 23 28 □
```

▪ Para las entradas de generación manual 2:

*P2D34354567.EXE*

```
Se mostrara el vector al principio y en cada vuelta del bucle
externo del metodo de ordenacion
20 17 13 28 14 23 15 42
13 20 17 14 28 15 23 42
13 14 20 17 15 28 23 42
13 14 15 20 17 23 28 42
13 14 15 17 20 23 28 42
13 14 15 17 20 23 28 42
```

*P2D4564567.EXE*

```
0
0
0
0
0
0
0
0
0
0
0
0
0
13
14
15
17
20
23
28
42
```

*P2D56347845.EXE*

```
0 15 23 14 28 13 17 20 2 □ 0 15 14 23 13 17 20 2 28 □ 0 14 15 13 17 20 2 23 28 □ 0
14 13 15 17 2 20 23 28 □ 0 13 14 15 2 17 20 23 28 □ 0 13 14 2 15 17 20 23 28 □ 0 13
2 14 15 17 20 23 28 □ 0 2 13 14 15 17 20 23 28 □
```

▪ Para las entradas de generación manual 3:

*P2D34354567.EXE*

```
Se mostrara el vector al principio y en cada vuelta del bucle
externo del metodo de ordenacion
2 137 1 28 45 23 15 42
1 2 137 15 28 45 23 42
1 2 15 137 23 28 45 42
```

*P2D4564567.EXE*

```
0
0
0
0
0
0
0
0
0
0
0
0
0
1
2
15
23
28
42
45
137
```

*P2D56347845.EXE*

```
0 15 23 42 28 1 45 2 2 □ 0 15 23 28 1 42 2 2 45 - 0 15 23 1 28 2 2 42 45 - 0 15 1 23 2 2
28 42 45 - 0 1 15 2 2 23 28 42 45 - 0 1 2 2 15 23 28 42 45 - 0 1 2 2 15 23 28 42 45 - 0 1
2 2 15 23 28 42 45 -
```

### 6.3.1.4.- Ejecución del *.exe* y comparación con la compilación y posterior ejecución de dicha práctica

Puesto que para está práctica ningún alumno ha dejado el *.exe* de su práctica esta opción no tiene sentido.

### 6.3.1.5.- Comparación de los resultados con los del profesor

- *6.3.1.5.a.- Comparación con la práctica del profesor*. Puesto que no se disponía de la práctica del profesor, no se puede realizar este tipo de comparación

- *6.3.1.5.b.- Ejecución de un programa para obtener los resultados*. Puesto que no se disponía de ningún programa, no se puede realizar este tipo de comparación

.

- *6.3.1.5.c.- Los resultados se obtendrán de ficheros*. Para realizar algún tipo más de prueba, hemos decidido utilizar esta opción, seleccionando como ficheros con los que se realizará la comparación, los ficheros de salida que genera una de las prácticas. Esto nos permitirá por lo tanto comparar las salidas entre los alumnos. Cogemos como referencia las salidas que generó durante las ejecuciones el alumno *P2D34354567* y los resultados de la comparación son los siguientes:

```
--  RESULTADO DE LA COMPARACIÓN DE LOS .EXE GENERADOS CON FICHEROS DE SALIDA ESPECÍFICOS  --

-------------------------------------------------------------------------
Nombre de la práctica       Tiene exe              Comparación

P2D4564567.pas              SI

- Entradas en el directorio C:\arma\Burbuja\GM_Entradas\GM_Entrada1

**          Fichero          de          resultados          seleccionado:
C:\ARMA\BURBUJA\COMPILADAS\EXEGENERADOS\GM_SALIDA1\P2D34354567\S1.TXT **

     - Fichero de salida: S1.TXT
                                   DISTINTOS

- Entradas en el directorio C:\arma\Burbuja\GM_Entradas\GM_Entrada2

**          Fichero          de          resultados          seleccionado:
C:\ARMA\BURBUJA\COMPILADAS\EXEGENERADOS\GM_SALIDA2\P2D34354567\S1.TXT **

     - Fichero de salida: S1.TXT
                                   DISTINTOS

- Entradas en el directorio C:\arma\Burbuja\GM_Entradas\GM_Entrada3

**          Fichero          de          resultados          seleccionado:
C:\ARMA\BURBUJA\COMPILADAS\EXEGENERADOS\GM_SALIDA3\P2D34354567\S1.TXT **

     - Fichero de salida: S1.TXT
                                   DISTINTOS

-------------------------------------------------------------------------
```

Nombre de la práctica Tiene exe Comparación

P2D56347845.pas SI

- Entradas en el directorio C:\arma\Burbuja\GM_Entradas\GM_Entrada1

** Fichero de resultados seleccionado: C:\ARMA\BURBUJA\COMPILADAS\EXEGENERADOS\GM_SALIDA1\P2D34354567\S1.TXT **

- Fichero de salida: S1.TXT
DISTINTOS

- Entradas en el directorio C:\arma\Burbuja\GM_Entradas\GM_Entrada2

** Fichero de resultados seleccionado: C:\ARMA\BURBUJA\COMPILADAS\EXEGENERADOS\GM_SALIDA2\P2D34354567\S1.TXT **

- Fichero de salida: S1.TXT
DISTINTOS

- Entradas en el directorio C:\arma\Burbuja\GM_Entradas\GM_Entrada3

** Fichero de resultados seleccionado: C:\ARMA\BURBUJA\COMPILADAS\EXEGENERADOS\GM_SALIDA3\P2D34354567\S1.TXT **

- Fichero de salida: S1.TXT
DISTINTOS

---
Nombre de la práctica Tiene exe Comparación

P2D34354567.pas SI

- Entradas en el directorio C:\arma\Burbuja\GM_Entradas\GM_Entrada1

** Fichero de resultados seleccionado: C:\ARMA\BURBUJA\COMPILADAS\EXEGENERADOS\GM_SALIDA1\P2D34354567\S1.TXT **

- Fichero de salida: S1.TXT
IGUALES

- Entradas en el directorio C:\arma\Burbuja\GM_Entradas\GM_Entrada2

** Fichero de resultados seleccionado: C:\ARMA\BURBUJA\COMPILADAS\EXEGENERADOS\GM_SALIDA2\P2D34354567\S1.TXT **

- Fichero de salida: S1.TXT
IGUALES

- Entradas en el directorio C:\arma\Burbuja\GM_Entradas\GM_Entrada3

** Fichero de resultados seleccionado: C:\ARMA\BURBUJA\COMPILADAS\EXEGENERADOS\GM_SALIDA3\P2D34354567\S1.TXT **

- Fichero de salida: S1.TXT
IGUALES

Se puede observar que todas las salidas son diferentes, pues como se ha podido ver anteriormente los tres alumnos muestran la salida con distintos formatos; evidentemente la única salida que coincide es la del alumno *P2D34354567*, puesto que fueron sus salidas las que cogimos como referencia.

#### 6.3.1.6.- Comparación de las estructuras de los programas

Además de no tener la práctica del profesor para realizar las comparaciones se tiene que para esta práctica se utilizan vectores y por lo tanto no es posible realizar la comparación de las estructuras, ya que la aplicación solo trata tipos simples de datos (excluyendo en su tratamiento, por tanto, los vectores).

### 6.3.2.- Generación automatica.

#### 6.3.2.1.- Ejecución de los *.exe* de los alumnos

Puesto que para está práctica ningún alumno ha dejado el *.exe* de su práctica esta opción no tiene sentido.

#### 6.3.2.2.- Ejecución de la práctica de un único alumno con datos de entrada específicos

Esta opción no esta disponible para la generación automática de los datos.

#### 6.3.2.3.- Compilación y ejecución de los programas

Se compilaron las 3 prácticas de los alumnos obteniéndose el siguiente resultado (el resultado de la compilación es el mismo que cuando se realizo la generación manual, ya que el tipo de datos de entrada seleccionado, manuales o automáticos, no influyen en la compilación) :

```
-- RESULTADO COMPILACIÓN --

COMPILA: P2D56347845.pas

COMPILA: P2D34354567.pas

COMPILA: P2D4564567.pas
```

Es decir compilaron el 100 % de las prácticas entregadas por los alumnos.

Se ejecutaron los tres ejecutables generados tras la compilación produciéndose los siguientes resultados:

```
-- RESULTADO EJECUCIÓN --


--------------------------------------------------------------------------------
- RESULTADOS PARA LOS DATOS DE ENTRADA QUE ESTÁN EN EL DIRECTORIO:
C:\arma\Burbuja\Compiladas\ExeGenerados\GA_Entradas\ Y QUE EMPIEZAN POR GA1

EJECUTA: P2D34354567.EXE
LA SALIDA QUE GENERA LA EJECUCIÓN SE PUEDE ENCONTRAR EN
C:\arma\Burbuja\Compiladas\ExeGenerados\GA_Salida1\P2D34354567\

EJECUTA: P2D4564567.EXE
LA SALIDA QUE GENERA LA EJECUCIÓN SE PUEDE ENCONTRAR EN
C:\arma\Burbuja\Compiladas\ExeGenerados\GA_Salida1\P2D4564567\

EJECUTA: P2D56347845.EXE
LA SALIDA QUE GENERA LA EJECUCIÓN SE PUEDE ENCONTRAR EN
C:\arma\Burbuja\Compiladas\ExeGenerados\GA_Salida1\P2D56347845\


--------------------------------------------------------------------------------
- RESULTADOS PARA LOS DATOS DE ENTRADA QUE ESTÁN EN EL DIRECTORIO:
C:\arma\Burbuja\Compiladas\ExeGenerados\GA_Entradas\ Y QUE EMPIEZAN POR GA2

EJECUTA: P2D34354567.EXE
LA SALIDA QUE GENERA LA EJECUCIÓN SE PUEDE ENCONTRAR EN
C:\arma\Burbuja\Compiladas\ExeGenerados\GA_Salida2\P2D34354567\

EJECUTA: P2D4564567.EXE
LA SALIDA QUE GENERA LA EJECUCIÓN SE PUEDE ENCONTRAR EN
C:\arma\Burbuja\Compiladas\ExeGenerados\GA_Salida2\P2D4564567\

EJECUTA: P2D56347845.EXE
LA SALIDA QUE GENERA LA EJECUCIÓN SE PUEDE ENCONTRAR EN
C:\arma\Burbuja\Compiladas\ExeGenerados\GA_Salida2\P2D56347845\
```

Es decir, ejecutan el 100 % de las prácticas.

Las salidas que genera cada práctica que sí ejecuta son las siguientes:

- Para las entradas de <u>generación automática 1</u>:

*P2D34354567.EXE*

```
Se mostrara el vector al principio y en cada vuelta del bucle
externo del metodo de ordenacion
6 -4 80 -195 -170 21 8 84 41 -133 0
-195 6 -4 80 -170 -133 21 8 84 41 0
-195 -170 6 -4 80 -133 0 21 8 84 41
-195 -170 -133 6 -4 80 0 8 21 41 84
-195 -170 -133 -4 6 0 80 8 21 41 84
-195 -170 -133 -4 0 6 8 80 21 41 84
-195 -170 -133 -4 0 6 8 21 80 41 84
```

*P2D4564567.EXE*

```
-195
-170
-133
-4
0
0
0
0
0
0
0
0
0
0
0
6
8
21
41
80
84
```

*P2D56347845.EXE*

```
0 -133 0 41 8 21 -170 -195 80 -4 6 2 □ -133 0 0 8 21 -170 -195 41 -4 6 2 80 P -133 0 0
8 -170 -195 21 -4 6 2 41 80 P -133 0 0 -170 -195 8 -4 6 2 21 41 80 P -133 0 -170 -195 0
-4 6 2 8 21 41 80 P -133 -170 -195 0 -4 0 2 6 8 21 41 80 P -170 -195 -133 -4 0 0 2 6 8
21 41 80 P -195 -170 -133 -4 0 0 2 6 8 21 41 80 P -195 -170 -133 -4 0 0 2 6 8 21 41 80
P -195 -170 -133 -4 0 0 2 6 8 21 41 80 P -195 -170 -133 -4 0 0 2 6 8 21 41 80 P
```

▪ Para las entradas de generación automática 2:

*P2D34354567.EXE*

```
Se mostrara el vector al principio y en cada vuelta del bucle
externo del metodo de ordenacion
-152 -137 -59 53 -199 -4 -5 -17 148 -167 0
-199 -152 -137 -59 53 -167 -4 -5 -17 148 0
-199 -167 -152 -137 -59 53 -17 -4 -5 0 148
-199 -167 -152 -137 -59 -17 53 -5 -4 0 148
```

*P2D4564567.EXE*

```
-199
-167
-152
-137
-59
-17
-5
-4
0
0
0
0
0
0
0
0
0
0
0
53
148
```

*P2D56347845.EXE*

```
0 -167 0 -17 -5 -4 -199 53 -59 -137 -152 2 □ -167 0 -17 -5 -4 -199 0 -59 -137 -152 2 53
5 -167 -17 -5 -4 -199 0 -59 -137 -152 0 2 53 5 -167 -17 -5 -199 -4 -59 -137 -152 0 0 2
53 5 -167 -17 -199 -5 -59 -137 -152 -4 0 0 2 53 5 -167 -199 -17 -59 -137 -152 -5 -4 0 0
2 53 5 -199 -167 -59 -137 -152 -17 -5 -4 0 0 2 53 5 -199 -167 -137 -152 -59 -17 -5 -4 0
0 2 53 5 -199 -167 -152 -137 -59 -17 -5 -4 0 0 2 53 5 -199 -167 -152 -137 -59 -17 -5 -4
0 0 2 53 5 -199 -167 -152 -137 -59 -17 -5 -4 0 0 2 53 5
```

### 6.3.2.4.- Ejecución del *.exe* y comparación con la compilación y posterior ejecución de dicha práctica

Puesto que para está práctica ningún alumno ha dejado el ejecutable de su práctica esta opción no tiene sentido.

### 6.3.2.5.- Comparación de los resultados con los del profesor

Puesto que para esta práctica no se disponía de la solución del profesor esta opción no tiene sentido.

### 6.3.2.6.- Comparación de las estructuras de los programas

Además de no tener la práctica del profesor para realizar las comparaciones se tiene que para esta práctica se utilizan vectores y por lo tanto no es posible realizar la comparación de las estructuras, ya que la aplicación solo trata tipos simples de datos (excluyendo en su tratamiento, por tanto, los vectores).

# 7.- APÉNDICE I: CÓDIGO DE LAS PRÁCTICAS

## 7.1.- ASCII

*P3D213564.pas*

```
PROGRAM Codigo (input,output);

    CONST
      Espacio=3;
    VAR
      x:char;
      fila,n,max:integer;
      s:text;
    BEGIN
      assign(s, 'Salida.txt');

      rewrite(s);


      write(s,'Fila 1');
      max:=255;
      FOR n:=1 TO max DO
        BEGIN
         IF n MOD 20 <>0 THEN
            BEGIN
              x:=chr(n);
              write(s,x:Espacio);
            END
         ELSE
            BEGIN
               writeln(s,x:Espacio);
               write(s,'Fila ',n div 20 + 1);
            END;
        END;
        close(s);

    END.
```

*P3D324532.pas*

```
PROGRAM ASCITABLA (input,output);
{Uses crt;}
VAR
k:integer;
s:text;
BEGIN
assign(s,'salida.txt');
rewrite(s);
 {clrscr;}
  FOR k:=0 TO 225 DO
  BEGIN
   BEGIN
```

```
    IF (k MOD 20=0) THEN
      BEGIN
       writeln(s);
       write(s,'fila',k DIV 20,' ');
      END
   END;
    write(s,chr(k),' ');
   END ;
close(s);
END.
```

*P3D346778.pas*

```
program caracter;
{uses crt;}
var i,j,z:integer;
S:TEXT;
begin
assign(s,'salida.txt');
rewrite(s);
{clrscr;}
j:=0;
z:=1;
write (s,'Fila 1 ');
    for i:=0 to 255 do
       begin
            write (s,chr(i));
            write (s,' ':2);
            j:=j+1;
            if j>20 then
            begin
              writeln(s);
              z:=z+1;
              write (s,'fila  ',z,' ':2);
              j:=0;
            end;
       end;
close(s);
end.
{program caracter;  {otra forma de hacerlo
var i,j:integer;

begin
j:=0;
    for i:=0 to 255 do
       begin
            write (chr(i));
            if (i mod 20=0) then
              writeln;
           end;
readln;
end. }
```

*P3D1347678.pas*

```
PROGRAM TablaCodigoASCII;
{uses crt;}
CONST
     ultimonum=255;
VAR  num:integer;
    letra: char;
    s:text;
BEGIN
{ clrscr;}
     Assign(s,'salida.txt');
     Rewrite(s);
     FOR num:=0 TO ultimonum DO
      BEGIN  letra:= chr(num);
            write(letra,' ');
            IF num mod 20=0 THEN
            BEGIN
              writeln;
              write('Fila',num div 20)
            END;
      END;
      close(s);
END.
```

*P3D1354647.pas*

```
PROGRAM NumASCII;
{USES crt;}
VAR
  i:integer;
  c:char;
  s:text;
BEGIN
assign(s,'salida.txt');
rewrite(s);
{clrscr;}
  FOR i:=0 TO 255 DO
     BEGIN
     c:= chr(i);
     write (s,c);
     IF i mod 20= 0 THEN
        writeln(s);
     END;
close(s);
END.
```

*P3D1425678.pas*

```
PROGRAM tabla_ASCII (input,output);
{USES
   crt;}
VAR
  j,i:integer;
  caractercodigo: char;
  s:text;
BEGIN
    assign(s,'salida.txt');
    rewrite(s);
  {  clrscr;}
    j:=255;
    FOR i:=0 TO j DO
       BEGIN
           caractercodigo:= chr(i);
           Write(s,caractercodigo:4);
           IF  i MOD 20=0 THEN
           Writeln(s);
       END;
END.
```

*P3D1456787.pas*

```
PROGRAM codigoascii (input,output);
{uses Crt;}
VAR caracter:char;
   codigo,fila:integer;
   s:text;
BEGIN
   assign(s,'salida.txt');
   rewrite(s);
   {ClrScr;}
   fila:=0;
   FOR codigo:=1 TO 255 DO
      BEGIN
          caracter:=chr(codigo);
          IF (codigo MOD 20 = 1) THEN
              begin fila:=fila + 1;
                   write(s,'fila ',fila:2,' ');
              end;
          IF (codigo MOD 20 = 0) THEN
              writeln(s,caracter,' ')
          ELSE write(s,caracter,' ');
      END;
   close(s);
END.
```

*P3D4564567.pas*

```
PROGRAM AscII (input,output);
{uses crt;}
VAR  i:integer;
     c:char;
     s:text;
BEGIN
  {clrscr;}
  assign(s,'s1.txt');
  rewrite(s);
  write (s,'fila1  ');
  FOR i:=1 TO 255 DO
    BEGIN
      IF i MOD 20=0 THEN
        BEGIN
          writeln(s);
          write (s,'fila',(i DIV 20)+1,'  ');
        END;
      c:=chr(i);
      write(s,c,'  ');
    END;
  close(s)
END.
{PROGRAM Rectangulo (input,output);
uses crt;
var n1,n2,i,k:integer;
BEGIN
  clrscr;
  writeln ('escriba los valores de la base y de la altura');
  readln (n1,n2);
  FOR i:=1 to n2 DO
    BEGIN
      FOR k:=1 TO n1 DO
        write('*');
      writeln
    END;
END.}
```

*P3D23432627.pas*

```
program Ascii;
{uses crt;}
var
 i,j:integer;
 S:TEXT;
 {i numero de elementos que hay,j nuemro de columnas}
Begin
 ASSIGN(s,'salida.txt');
 rewrite(s);
  {clrscr;}
   j:=1;
   for i:=0 to 255 do
    begin
      if i mod 20 = 0 then
```

```
        begin
         writeln(s);
         write(s,'fila',j,' ');
         j:=j + 1;
        end;
       write(s,chr(i),' ');

     end;
    close(s);
End.
```

*P3D34354567.pas*

```
PROGRAM Tabla_ASCII (input, output);
{uses Crt;}
CONST
    UltimNumero = 255;
VAR
    Num: Integer;
    Caracter: char;
    S:Text;
BEGIN
  {  ClrScr;}
    assign(s,'s1.txt');
    rewrite(s);
    FOR Num:= 0 TO UltimNumero DO
       BEGIN
       Caracter:= chr(Num);
       Write(s,caracter,'  ');
       IF ((Num MOD 20) = 0) THEN
          BEGIN
          Writeln(s);
          Write(s,'Fila ',(Num DIV 20) +1,'   ',caracter);
          END;{IF}
       END;{FOR}
    close(s);
END.
```

*P3D54546765.pas*

```
program Ascii;
{uses crt;}
var
 i,j:integer;
 s:text;
 {i numero de elementos que hay,j nuemro de columnas}
Begin
  {clrscr;}
  assign(s,'s1.txt');
  rewrite(s);
    j:=1;
    for i:=0 to 255 do
```

```
      begin
       if i mod 20 = 0 then
        begin
         writeln(s);
         write(s,'fila',j,' ');
         j:=j + 1;
        end;
      write(s,chr(i),' ');

     end;
    close(s);
  {  readln;}
End.
```

*P3D56347845.pas*

```
 PROGRAM TablaASCII (Input,output);
{ uses crt;}
 VAR
   i : Integer;
   s:text;

 BEGIN
   { clrscr;}
    assign(s,'s1.txt');
    rewrite(s);
    FOR i := 0 TO 255 DO
     BEGIN
       IF (i MOD 20 = 0) THEN
         BEGIN
          writeln(s);
          write(s,'Fila ',i DIV 20 + 1)
         END;
        write(s,'  ',chr(i));
     END;
    close(s);
{readln;}
 END.
```

*P3PROF.pas*

```
PROGRAM CodigoASCII (input,output);
{uses crt;}
CONST TamFila = 10;
VAR i,j : Integer;
s:text;
BEGIN
     {clrscr;}
     assign(s,'Salida.txt');

     rewrite(s);
```

```
    write(s,'Fila 1.  ');
    FOR i := 1 TO 256 DO
    BEGIN
     write(s,char(i):3);
     IF i MOD 20 = 0 THEN
          BEGIN
            writeln(s);
            write(s,'Fila ',i DIV 20+1,'. ');
          END
    END;
    close(s);

END.
```

## 7.2.- MCD

*P1D1347678.pas*

```
Program euclideos (input, output);
Uses crt;
Var
  num1, num2: Integer;
  s,e:text;
Function mcd (n1, n2: Integer): integer;
Var
  resto: Integer;
Begin
  while n2 <> 0 do
    Begin
      resto:= n1 Mod n2;
      n1:= n2;
      n2:= resto;
    End;
    mcd:= n1
End;

Begin{programa principal}
  assign(s,'salida.txt');
  assign(e,'e1.txt');
  reset(e);
  rewrite(s);
  clrscr;
  writeln ('introduzca dos valores enteros');
  readln (e,num1, num2);
  writeln (s,'el maximo comun divisor de ', num1, ' y ', num2, ' es ', mcd(num1, num2));
  close(s);
  close(e);
End.
```

*P1D1354647.pas*

```
PROGRAM metodo_euclides;
VAR
  a,b: integer;
  s,e:text;
BEGIN
    assign(s,'salida.txt');
    assign(e,'e1,txt');
    rewrite(s);
    reset(e);

    writeln(s,'Introduzca el primer n£mero');
    readln(e,a);
    writeln(s,'Introduzca el segundo n£mero');
    readln(e,b);
    WHILE a<>b DO
      BEGIN
      IF a>b THEN
        a:= a-b
      ELSE
        b:= b-a;
      END;
    writeln(s,'El MCD es: ',a);
    close(s);
    close(e);
END.
```

*P1D1425678.pas*

```
{program max_com_div;
var x,y:integer;
begin
     writeln ('Escribe dos n£meros para hallar el mcd');
     readln (x,y);
     while x<>y do
         begin
             if x>y then
               x:=x-y
             else
               y:=y-x
         end;
     writeln ('El mcd es: ',x);
end. }
program max_com_div;{otra forma}
var x,y:integer;
s,e:text;
function mcd (a,b:integer):integer;
begin
   if a=b then mcd:=a
   else if a>b then mcd:=mcd (b,a-b)
   else if b>a then mcd:=mcd (a,b-a)
end;
begin
```

```
    assign(s,'salida.txt');
    assign(e,'e1.txt');
    rewrite(s);
    reset(e);
    writeln (s,'Escribe dos n£meros para hallar el mcd');
    readln (e,x,y);
    writeln (s,'El mcd es: ',mcd(x,y));
    close(s);
    close(e);
end.
```

*P1D213564.pas*

```
PROGRAM maxcomundivisor (input,output);
USES Crt;
VAR     n1,n2,n3:integer;
s,e:text;
FUNCTION maximo (a,b:integer):integer;    {c lculo de m ximo}
BEGIN
 WHILE a<>b DO
    BEGIN
      IF a>b THEN
      a:=a-b
      ELSE
      b:=b-a;
    END;
 maximo:=a;

END;

BEGIN      {programa principal}
   assign(s,'salida.txt');
   assign(e,'e1.txt');
   rewrite(s);
   reset(e);
   ClrScr;
   writeln(s,'introduzca los dos valores');
   readln(e,n1,n2);
   writeln(s,'El m ximo com£n divisor de',n1:4,n2:4,' es =',maximo(n1,n2));
   close(s);
   close(e);
END.
```

*P1D23432627.pas*

```
PROGRAM MaximoComDivisor;
uses crt;
var num1,num2:integer;
s,e:text;
{OTRA FORMA}

{ function mcd(num1,num2: integer):integer;
  begin if num1=num2 then mcd:= num1
      else if num1<num2 then mcd:=mcd(num1,num2-num1)
      else if num2<num1 then mcd:= mcd(num2,num1-num2);
   end;
 }
FUNCTION mcd(num1,num2:integer):integer;

BEGIN WHILE (num1<>num2) DO
       Begin IF num1<num2 THEN
             num2:=num2-num1
            ELSE
              num1:= num1-num2
       End ;
       mcd:=num1;
END;

BEGIN
assign(s,'salida.txt');
assign(e,'e1.txt');
reset(e);
rewrite(s);
clrscr;
      writeln(s,'Escriba los dos numeros');
      readln(e,num1,num2);
      writeln(s,'El M.C.D. de ',num1:4,num2:4,' es ',mcd(num1,num2));
      close(s);
      close(e);
END.
```

*P1D34354567.pas*

```
{21 Enero 2003}

PROGRAM CalculoMCD (input, output);
uses Crt;
VAR
  Numero1, Numero2,salida: Integer;
  entrada, resul:Text;
{RECURSIVIDAD}
FUNCTION MCD(Num1, Num2: Integer):Integer;
  BEGIN
  IF Num1>Num2 THEN MCD:=MCD(Num1-Num2, Num2)
  ELSE IF Num1 < Num2 THEN MCD:= MCD(Num1, Num2 - Num1)
  ELSE MCD:= Num1
  END;
```

```
BEGIN
  ClrScr;
  Writeln('Introduce los dos numeros de los que se quiere el m.c.d:');
  Assign(Entrada,'E1.txt');
  Reset(Entrada);
  read(Entrada,numero1,numero2);
  salida:=MCD(Numero1, Numero2);
  Assign(resul,'S1.txt');
  rewrite(resul);
  write(resul,salida);
  close(entrada);
  close(resul);

END.

{BUCLE}
{WHILE Numero1 <> Numero2 DO
    BEGIN
    IF Numero1> Numero2 THEN Numero1:= Numero1 - Numero2
    ELSE Numero2:=Numero2 - Numero1
    END;
 Writeln(' El mcd es',Numero1);}
```

*P1D346778.pas*

```
PROGRAM maximocomundivisor (input,output);
Uses crt;
VAR
f,n1,n2:integer;
s,e:text;
FUNCTION mcd (x,y:integer):integer;
     BEGIN
       WHILE x<>y DO
         IF x>y THEN
           x:=x-y
         ELSE
           y:=y-x;
       mcd:=x;
     END;

BEGIN{programa principal}
assign(s,'salida.txt');
assign(e,'e1.txt');
rewrite(s);
reset(e);
clrscr;
writeln(s,'Introduzca una lista de numeros separados por blancos y acabados en 0 0');
read(e,n1,n2);
WHILE (n1<>0) AND (n2<>0) DO
BEGIN
f:=mcd(n1,n2);
writeln(s,'El mcd de ',n1,' y ',n2,' es ',f);
```

```
read(e,n1,n2);
END;
close(s);
close(e);

END.
```

*P1D54546765.pas*

```
program euclides ;
uses crt;
var
 numero1,numero2, salida:integer;
 entrada, resul:Text;

function mcd (n1,n2:integer):integer;
  var resto:integer;
  begin
       while n2 <> 0 do
        begin
          resto:= n1 mod n2;
          n1:=n2;
          n2:=resto;
        end;
        mcd:=n1;

  end;
begin
     clrscr;
     writeln('introduzca dos numeros para hallar el maximo comun divisor');
     Assign(Entrada,'E1.txt');
  Reset(Entrada);
  read(Entrada,numero1,numero2);
  salida:=MCD(Numero1, Numero2);
  Assign(resul,'S1.txt');
  rewrite(resul);
  write(resul,salida);
  close(entrada);
  close(resul);

     writeln('el mcd de ',numero1,' y ', numero2,' es ',mcd(numero1,numero2));
end.
```

*P1D56347845.pas*

```
PROGRAM CalculoMaxComunDiv (Input,output);
uses crt;
VAR
  N1, N2, salida :Integer;
  entrada, resul : Text;

  FUNCTION mcd ( A, B :Integer ) :Integer;
  VAR
    Resto: Integer;
  BEGIN
    WHILE B <> 0 DO
      BEGIN
        Resto := A MOD B;
        A := B;
        B:= Resto;
      END;
    mcd := A;
  END;

BEGIN {****************** programa principal **********************}
    clrScr;
    writeln;
    writeln('Este programa calcula el m ximo com£n divisor',
            ' de dos n£meros enteros positivos.');
    writeln;
    write('Escriba el par de n£meros (el mayor en primer lugar). ');
  Assign(Entrada,'E1.txt');
  Reset(Entrada);
  read(Entrada,n1,n2);
  salida:=MCD(N1, N2);
  Assign(resul,'S1.txt');
  rewrite(resul);
  write(resul,salida);
  close(entrada);
  close(resul);
 END.
```

*P1D78679540.pas*

```
PROGRAM AlgoritmoDeEuclides (input,output);
 uses crt;
 VAR numero1,numero2,r, SALIDA:Integer;
 ENTRADA, RESUL:Text;
 FUNCTION mcd (x,y:Integer):Integer;
     BEGIN
       r:= x MOD y;
       WHILE r<>0 DO
         BEGIN
           x:=y;
           y:=r;
           r:= x MOD y
```

```
        END;
      mcd:=y
    END;
BEGIN
   clrscr;
       Assign(Entrada,'E1.txt');
 Reset(Entrada);
 read(Entrada,numero1,numero2);
 salida:=MCD(Numero1, Numero2);
 Assign(resul,'S1.txt');
 rewrite(resul);
 write(resul,salida);
 close(entrada);
 close(resul);

END.
```

## 7.3.- BURBUJA

*P2D34354567.pas*

```
PROGRAM MetodoBurbuja(input,output);
CONST
  MaxElem = 25;
TYPE
  Vector = ARRAY[1..MaxElem] OF Integer;
VAR
  V: Vector;
  ArchE, salida: Text;
  NumElem: Integer;

PROCEDURE Almacenar(VAR Archivo: text;VAR V2: Vector;
                VAR Contador:Integer);
VAR
  Num: Integer;
BEGIN
  Reset(Archivo);
  Contador:= 0;
  WHILE NOT eof(Archivo) DO
  BEGIN
  Read(Archivo,Num);
  Contador:= Contador +1;
  V2[Contador]:= Num
  END;
  Close(Archivo)
END;
```

```
PROCEDURE MostrarVector(VAR V2: vector;Contador: Integer; VAR salida:Text);
VAR
 i: Integer;
BEGIN
 FOR i:= 1 TO Contador DO Write(salida,V2[i],' ');
 writeln(salida);
END;

PROCEDURE Ordenar(VAR V2:vector;contador:Integer;VAR salida:text);
VAR
  i,j,Aux: Integer;
  Ord:Boolean;
BEGIN
  ord:= FALSE;
  i:= 1;
  WHILE (i<= Contador) AND NOT ord DO
  BEGIN
  FOR j:= Contador DOWNTO i+1 DO
    BEGIN
     ord:= TRUE;
     IF V2[j] < V2[j-1] THEN
       BEGIN
       Aux:= V2[j];
       V2[j]:= V2[j-1];
       V2[j-1]:= Aux;
       ord:= FALSE;
       END
    END;
  MostrarVector(V2,Contador,salida);
  i:= i+1
  END;
END;

BEGIN
  Assign(ArchE,'E1.txt');
  Assign(salida,'S1.txt');
  rewrite(salida);
  Writeln(salida,'Se mostrara el vector al principio y en cada vuelta del bucle ');
  Writeln(salida,'externo del metodo de ordenacion');
  Almacenar(ArchE,V,NumElem);
  MostrarVector(V,NumElem,salida);
  Ordenar(V,NumElem,salida);
  close(salida);
END.
```

*P2D4564567.pas*

```
PROGRAM Burbuja (input,output);
 CONST MaxElem=20;
 TYPE Vector=ARRAY[1..MaxElem] OF integer;
 VAR Aent, salida:text;
    x,numelem:integer;
    V:Vector;

 PROCEDURE Leer(VAR Aent:text;VAR V:Vector; VAR numelem:integer);
 VAR n,i:integer;
 BEGIN
  reset(Aent);
  numelem:=1;
  WHILE NOT eof(Aent) DO
  BEGIN
    FOR i:=1 TO MaxElem DO
    BEGIN
      read(Aent,n);
      V[i]:=n;
      numelem:=numelem+1
    END;
  END;
  close(Aent);
 END;
 PROCEDURE Ordenar (VAR V:Vector; numelem:integer);
 VAR j,k,aux:integer;
 BEGIN
           FOR j:=1 TO numelem DO
  BEGIN
    FOR k:=numelem DOWNTO 2 DO
    BEGIN
     IF V[k]<V[k-1] THEN
     BEGIN
       aux:=V[k-1];
       V[k-1]:=V[k];
       V[k]:=aux;
     END;
     END;
  END;
 END;
 BEGIN
           assign(Aent,'E1.txt');
      assign(salida,'S1.txt');
      rewrite(salida);
  Leer(Aent,V,numelem);
  Ordenar(V,numelem);
  FOR x:=1 TO numelem DO
  writeln(salida,V[x]);
  close(salida);
 END.
```

*P2D56347845.pas*

```
program ordenacion;
type vector1= array[1..20] of integer;
var
vector:vector1;
  archivo, salida:text;
  tama:integer;

procedure inicializar (var v:vector1);
var i:integer;
begin
    for i:=1 to 20 do
      v[i]:=0;
end;

procedure leer(var v:vector1;var archi:text;var i:integer);
var n:integer;

begin
 while not eof(archi) do
     begin
     read(archi,n);
     v[i]:=n;
     i:=i+1;
     end;
end;

procedure burbuja (var v:vector1;tama:integer;var salida:text);
var i,j,aux:integer;
begin
    for i:=1 to tama do
    begin
        for  j:=tama downto 0 do
        begin
            if v[j] > v[j-1] then begin
                                          v[aux]:=v[j-1];
            v[j-1]:=v[j];
            v[j]:=v[aux];
            end;
         write(salida,v[j],' ')
        end;
    writeln(salida);
    end;
end;

begin {principal}
inicializar(vector);
assign(archivo,'E1.txt');
assign(salida,'S1.txt');
rewrite(salida);
reset(archivo);
leer(vector,archivo,tama);
burbuja(vector,tama, salida);
close(archivo);
close(salida);
end.
```

# 8.- LISTA DE PALABRAS CLAVE

## C


- Compilación..... 1, 17, 19, 20-22, 25, 29-30, 33-34, 40, 85, 87, 91, 98, 100, 102-103, 105, 108-109, 113, 115, 118
- Comparación
  - de resultados,..... 1, 17, 21-25, 27, 29-34, 64, 91, 93, 96, 102, 105, 113, 118
  - de estructuras de los programas..... 1, 35-36, 38, 68-69, 71, 75-76, 78-79, 98, 102, 105, 115, 118


## E


- Ejecución de programas.....1, 16-18, 20-23, 40, 45, 53, 64-65, 67, 80, 86, 91, 93, 100, 102-103, 105, 108, 113, 118


## G


- Generación
  - automática,..... 1, 12, 18, 21, 30, 44, 62, 80, 99, 103, 106, 107, 115-117
  - manual..... 1, 12, 15, 18, 21, 30-31, 40, 51, 63, 99, 100, 106-108, 110-112, 115


## L


- Listados.....1, 38-40, 48-51, 64


## P


- Pruebas.....19, 51, 80, 99, 106-107

# 9.- BIBLIOGRAFÍA


Sobre Pascal:

- Dale, N. & Weems C. "Pascal. Segunda edición". McGraw-Hill

Sobre Java:

- Eckel, B. "Thinking in Java. Second Edition". Prentice-Hall. 2000
- "JAVA 2. Manual de usuario y tutorial". 3ª Edición. RA-MA. 2002
- Web oficial de Java: http://www.sun.com/java

Sobre la comparación de la estructura de programas:

- Michael Luck & Mike Joy. (Department of Computer Science. University of Warwick)
  "A SECURE ON-LINE SUBMISSION SYSTEM"